



# **Grid-Based Semantic Integration of Heterogeneous Data Resources: Implementation on a HealthGrid**

**by**

**Aisha Naseer**

**A thesis submitted for the degree of Doctor of Philosophy**

**School of Information Systems, Computing and Mathematics  
Brunel University**

**July 2007**

## **Dedication**

**To my parents Naseer Ahmed and Faiza Naseer and my brother Mohsin Naseer.**

## **Acknowledgement**

Many thanks go to my supervisor Dr. Lampros K. Stergioulas, who has been fundamentally important during all phases of my research by providing me support, advice, guidance and encouragement.

Valuable comments and discussions from Prof. Zidong Wang are much appreciated and thanked. Moreover, I am particularly grateful to Dr. Maozhen Li for his fruitful suggestions and constructive remarks and to Suhel Hammound for his technical assistance

My sincere thanks also go to Prof. Ray Paul and Dr. Jasmina Zugic for introducing me to the art of thesis writing.

My special thanks go to Dr. Navonil Mustafee for all his support in thesis compilation and for technical assistance and to Marina Bogovac for her assistance in thesis writing.

## Abstract

The semantic integration of geographically distributed and heterogeneous data resources still remains a key challenge in Grid infrastructures. Today's mainstream Grid technologies hold the promise to meet this challenge in a systematic manner, making data applications more scalable and manageable. The thesis conducts a thorough investigation of the problem, the state of the art, and the related technologies, and proposes an *Architecture for Semantic Integration of Data Sources* (ASIDS) addressing the semantic heterogeneity issue. It defines a simple mechanism for the interoperability of heterogeneous data sources in order to extract or discover information regardless of their different semantics. The constituent technologies of this architecture include Globus Toolkit (GT4) and OGSA-DAI (Open Grid Service Architecture Data Integration and Access) alongside other web services technologies such as XML (Extensive Markup Language). To show this, the ASIDS architecture was implemented and tested in a realistic setting by building an exemplar application prototype on a HealthGrid (pilot implementation).

The study followed an empirical research methodology and was informed by extensive literature surveys and a critical analysis of the relevant technologies and their synergies. The two literature reviews, together with the analysis of the technology background, have provided a good overview of the current Grid and HealthGrid landscape, produced some valuable taxonomies, explored new paths by integrating technologies, and more importantly illuminated the problem and guided the research process towards a promising solution. Yet the primary contribution of this research is an approach that uses contemporary Grid technologies for integrating heterogeneous data resources that have semantically different data fields (attributes). It has been practically demonstrated (using a prototype HealthGrid) that discovery in semantically integrated distributed data sources can be feasible by using mainstream Grid technologies, which have been shown to have some significant advantages over non-Grid based approaches.



# Table of Contents

DEDICATION.....	II
ACKNOWLEDGEMENT .....	III
ABSTRACT.....	IV
TABLE OF CONTENTS.....	V
LIST OF FIGURES .....	VIII
LIST OF TABLES .....	X
LIST OF ACRONYMS.....	XI
AUTHOR'S DECLARATION.....	XV
CHAPTER 1 .....	1
1.1    MOTIVATION AND RESEARCH CHALLENGE.....	1
1.2    OVERVIEW OF THE PROBLEM AREA .....	2
1.3    RESEARCH AIMS & OBJECTIVES .....	4
1.4    CONTEXT OF THE EXEMPLAR - HEALTHGRIDS.....	6
1.5    RESEARCH APPROACH .....	7
1.6    THESIS OUTLINE .....	8
PART-I.....	12
CHAPTER 2 .....	13
2.1    INTRODUCTION .....	13
2.2    PRELIMINARY RESEARCH INTO GRID & WEB TECHNOLOGIES .....	13
2.2.1    Grid Technologies (OGSA, OGSF, WSRF, GT4, OGSA-DAI).....	14
2.2.2    Web Services Technologies (XML, WSDL, UDDI, SOAP, SOI, WSDM) .....	17
2.3    CONVERGENCE OF GRID AND WEB SERVICES TECHNOLOGIES .....	19
2.4    CONCLUSION .....	25
CHAPTER 3 .....	27
3.1    INTRODUCTION .....	27
3.2    INTRODUCTION TO GRIDS .....	28
3.3    TYPES OF RESOURCES ON GRIDS.....	30
3.3.1    Physical Resources.....	31
3.3.2    Logical Resources .....	32
3.4    RESOURCE DISCOVERY ON GRIDS.....	35
3.4.1    The Need for Resource Discovery .....	36
3.4.2    Grid Resource Discovery Issues & Technical Limitations.....	37
3.4.3    Expected Benefits from Resource Discovery in Grids .....	38
3.5    TAXONOMY OF RESOURCE DISCOVERY METHODS .....	38
3.5.1    The Centralised Resource Discovery Model .....	41
3.5.2    The Distributed Resource Discovery Model.....	44
3.5.3    The Semi-Distributed Resource Discovery Model.....	61
3.6    CRITICAL ANALYSIS .....	63
3.7    CONCLUSION .....	66
CHAPTER 4 .....	68
4.1    INTRODUCTION .....	68
4.2    INTRODUCTION TO HEALTHGRIDS .....	70
4.3    HEALTHCARE NEEDS GRID TECHNOLOGY.....	72

4.4	TAXONOMY OF HEALTHGRIDS TYPES .....	75
4.4.1	<i>BioGrid</i> .....	76
4.4.2	<i>MediGrid</i> .....	79
4.4.3	<i>BioMedicalGrid</i> .....	84
4.4.4	<i>PharmaGrid</i> .....	85
4.4.5	<i>CareGrid</i> .....	89
4.5	TYPES OF RESOURCES ON HEALTHGRIDS .....	90
4.5.1	<i>Data, Information or File (DIF) Resources</i> .....	91
4.5.2	<i>Application &amp; Peripheral (AP) Resources</i> .....	95
4.5.3	<i>Service Resources</i> .....	96
4.5	RESOURCE DISCOVERY IN HEALTHGRIDS.....	97
4.5.1	<i>The Resource Discovery Challenge</i> .....	97
4.5.2	<i>Heterogeneity and Coding Issues</i> .....	98
4.5.3	<i>Resource Discovery Problems and Solutions</i> .....	100
4.6	CONCLUSION .....	102
<b>PART-II .....</b>		<b>105</b>
<b>CHAPTER 5 .....</b>		<b>106</b>
5.1	INTRODUCTION .....	106
5.2	RESEARCH APPROACH .....	106
5.3	RESEARCH DESIGN .....	107
5.3.1	<i>Phase I: Literature Survey</i> .....	109
5.3.2	<i>Phase II: Hypothesis Formulation</i> .....	109
5.3.3	<i>Phase III: System Architecture</i> .....	110
5.3.4	<i>Phase IV: Setting up an Exemplary Application Environment</i> .....	111
5.3.5	<i>Phase V: Prototyping</i> .....	111
5.3.6	<i>Phase VI: Evaluation</i> .....	112
5.3.7	<i>Phase VII: Conclusions</i> .....	112
5.4	CONCLUSION .....	113
<b>CHAPTER 6 .....</b>		<b>114</b>
6.1	INTRODUCTION .....	114
6.2	PROPOSED N-TIER-TO-N-TIER APPLICATION ARCHITECTURE .....	114
6.2.1	<i>Component-I: Physically Distributed Data Sources (PDDS)</i> .....	118
6.2.2	<i>Component-II: Semantic Query Engine (SQE)</i> .....	118
6.2.3	<i>Component-III: Web-based User Interface (WUI)</i> .....	121
6.3	CONCLUSION .....	121
<b>CHAPTER 7 .....</b>		<b>123</b>
7.1	INTRODUCTION .....	123
7.2	APPLICATION CONTEXT .....	123
7.3	PROTOTYPE DEVELOPMENT .....	125
7.3.1	<i>Setting up the Prototype Application Environment</i> .....	125
7.3.2	<i>Building the Prototype</i> .....	139
7.4	OPERATIONAL FLOW OF THE PROTOTYPE .....	143
7.5	CONCLUSIONS.....	145
<b>CHAPTER 8 .....</b>		<b>146</b>
8.1	INTRODUCTION .....	146
8.2	EXPERIMENT-I: TESTING OUT THE SYSTEM WITH A SINGLE GRID INSTALLATION.....	149
8.2.1	<i>Overview of Experiment-I</i> .....	149
8.2.2	<i>Objective of Experiment-I</i> .....	150
8.2.3	<i>Experiment-I Setup</i> .....	150
8.2.4	<i>Results Analysis of Experiment-I</i> .....	151
8.3	EXPERIMENT-II: TESTING OUT THE SYSTEM WITH MULTIPLE GRID INSTALLATIONS ..	155
8.3.1	<i>Overview of Experiment-II</i> .....	155

8.3.2	<i>Objective of Experiment-II</i> .....	156
8.3.3	<i>Experiment-II Setup</i> .....	156
8.3.4	<i>Results Analysis of Experiment-II</i> .....	157
8.4	CONCLUSIONS.....	162
<b>CHAPTER 9 .....</b>		<b>164</b>
9.1	INTRODUCTION .....	164
9.2	RESEARCH SUMMARY.....	164
9.3	CONCLUSIONS.....	166
9.4	FURTHER RESEARCH.....	169
<b>REFERENCES.....</b>		<b>170</b>
<b>APPENDIX-A.....</b>		<b>190</b>
<b>APPENDIX-B .....</b>		<b>192</b>

# List of Figures

FIGURE 1: WSRF SPECIFICATIONS.....	15
FIGURE 2: MAIN CATEGORIZATIONS OF GLOBUS TOOLKIT (GT4) ARCHITECTURE.....	16
FIGURE 3: WEB-BASED GRID INFRASTRUCTURE .....	21
FIGURE 4: IMPLEMENTATION ARCHITECTURE FOR GRID APPLICATIONS.....	22
FIGURE 5: WS-BASED RESOURCE DISCOVERY MODEL: HEALTHGRID EXAMPLE (THE LOW LEVEL SERVICES SHOWN IN THE MODEL ARE TAKEN FROM THE HEALTHGRID EXEMPLAR) .....	24
FIGURE 6: HIERARCHY OF GRID RESOURCES .....	31
FIGURE 7: TAXONOMY OF RESOURCE DISCOVERY METHODS.....	41
FIGURE 8: PROTOCOL MESSAGING BETWEEN SERVERS/USERS.....	46
FIGURE 9: TAXONOMY OF HEALTHGRIDS TYPES .....	75
FIGURE 10: A HIERARCHY OF HEALTHGRID RESOURCES .....	90
FIGURE 11: DIF RESOURCE TREE .....	91
FIGURE 12: PATIENT RECORDS DETAILS .....	92
FIGURE 13: HEALTHCARE STAFF DETAILS .....	93
FIGURE 14: DRUG DETAILS .....	93
FIGURE 15: DISEASES & OTHER BIOMED SCIENTIFIC INFORMATION .....	94
FIGURE 16: AP RESOURCE TREE.....	95
FIGURE 17: HEALTHGRID SERVICE RESOURCE TREE.....	97
FIGURE 18: RESEARCH DESIGN.....	108
FIGURE 19: N-TIER-TO-N-TIER ASIDS ARCHITECTURE.....	117
FIGURE 20: ONTOLOGY SPECIFICATION .....	120
FIGURE 21: GT4 SERVICES CONTAINER.....	126
FIGURE 22: INTERFACE TO DATA SOURCES .....	127
FIGURE 23: GUI FOR MYSQL ADMINISTRATOR .....	128
FIGURE 24: MQA GUI OF TABLE EDITOR SHOWING FIELDS FOR THE TABLE "DINFO_1" (DS1)...	129
FIGURE 25: MQA GUI OF TABLE EDITOR SHOWING FIELDS FOR THE TABLE "MEDINFO_1" (DS2) .....	130
FIGURE 26: MQA GUI OF TABLE EDITOR SHOWING FIELDS FOR THE TABLE "PHARMAINFO_1" (DS3) .....	130
FIGURE 27: SCREENSHOT OF THE MQB GUI INTERFACE THAT SHOWS VALUES IN THE TABLE "DINFO_1" (DS1).....	131
FIGURE 28: SCREENSHOT OF THE MQB GUI INTERFACE THAT SHOWS VALUES IN THE TABLE "MEDINFO_1" (DS2).....	132
FIGURE 29: SCREENSHOT OF THE MQB GUI INTERFACE THAT SHOWS VALUES IN THE TABLE "PHARMAINFO_1" (DS3).....	132
FIGURE 30: LIST OF DSRs EXPOSED VIA DATADiscovery1 .....	134
FIGURE 31: LOCATION OF THE DATARESOURCECONFIG.XML FILE FOR DS1 .....	135
FIGURE 32: DSR CONFIGURATION FILE (XML) FOR DS1 .....	136
FIGURE 33: SEMANTIC ONTOLOGY MATCHING FOR DS1 .....	137
FIGURE 34: SEMANTIC ONTOLOGY MATCHING FOR DS2.....	138
FIGURE 35: SEMANTIC ONTOLOGY MATCHING FOR DS3 .....	138
FIGURE 36: GUI USER INTERFACE (JSP PAGE) .....	139
FIGURE 37: APPLICATION RUNNING IN TOMCAT SERVER 5.0.28.....	140
FIGURE 38: XML STYLE SHEET TRANSFORM FILE USED TO TRANSFORM THE QUERY RESULTS FROM XML FORMAT INTO HTML FORMAT.....	142
FIGURE 39: SEMANTIC QUERY RESULTS .....	143
FIGURE 40: OPERATIONAL FLOW OF THE PROTOTYPE.....	144
FIGURE 41: EXAMPLES OF SQL USER QUERIES USED.....	148
FIGURE 42: SINGLE GI CONTAINING MULTIPLE DSS .....	149
FIGURE 43: DATA SETS IN DS3 .....	149
FIGURE 44: TESTING OUT THE SYSTEM WITH SINGLE GRID INSTALLATION .....	151
FIGURE 45: FLOWCHART FOR EXPERIMENT-I.....	152

FIGURE 46: MULTIPLE GIS CONTAINING ONE DS EACH..... 155

FIGURE 47: TESTING OUT THE SYSTEM WITH MULTIPLE GRID INSTALLATIONS ..... 158

FIGURE 48: FLOWCHART FOR EXPERIMENT-II ..... 159

FIGURE 49: TIME STAMPS FOR EXPERIMENT-II..... 161

FIGURE 50: GLOBUS WEBPAGE FROM WHERE GT4.0.4 WAS DOWNLOADED ..... 190

FIGURE 51: OGSA-DAI WEBPAGE FROM WHERE OGSA-DAI WSRF 2.2 WAS DOWNLOADED ..... 191

FIGURE 52: MYSQL WEBPAGE FROM WHERE MYSQL GUI TOOLS WERE DOWNLOADED ..... 191

## List of Tables

TABLE 1: A COMPARISON OF DIFFERENT SYSTEMS SURVEYED THAT TAXONOMISES VARIOUS SEMANTIC INTEGRATION MECHANISMS USED BY THEM .....	65
TABLE 2: RESOURCE DISCOVERY PROBLEMS AND POSSIBLE SOLUTIONS .....	101
TABLE 3: FIELDS OF THREE DATA SOURCES HAVING DIFFERENT NAMES BUT CONTAINING THE SAME TYPE OF INFORMATION .....	129
TABLE 4: HARDWARE AND SOFTWARE SPECIFICATIONS .....	147
TABLE 5: SETUP FOR EXPERIMENT-I .....	150
TABLE 6: ELAPSED TIME TO QUERY SINGLE GI WITH MULTIPLE DSS .....	153
TABLE 7: SHOWING OVERHEAD FOR SEMANTIC MAP GENERATION .....	154
TABLE 8: SETUP FOR EXPERIMENT-II .....	157
TABLE 9: ELAPSED TIME TO QUERY MULTIPLE GIS CONTAINING ONE DS EACH .....	160

## **List of Acronyms**

<b>3D-EM</b>	<b>Three-Dimensional Electron Microscopy</b>
<b>AJO</b>	<b>Abstract Job Objects</b>
<b>ANN</b>	<b>Artificial Neural Network</b>
<b>AP</b>	<b>Application and Peripheral resource</b>
<b>API</b>	<b>Application Programming Interface</b>
<b>ASIDS</b>	<b>Architecture for Semantic Integration of Data Sources</b>
<b>BIRN</b>	<b>Biomedical Information Research Network</b>
<b>C&amp;M</b>	<b>construction and maintenance</b>
<b>CAA</b>	<b>Computational Aero Acoustics</b>
<b>CAN-DHT</b>	<b>Content Address Network-based Distributed Hash Table</b>
<b>CFD</b>	<b>Computational Fluid Dynamics</b>
<b>CHK</b>	<b>Content-Hash Key</b>
<b>CPU</b>	<b>Central Processing Unit</b>
<b>CSP</b>	<b>Change Sensitive Protocol</b>
<b>DAG</b>	<b>Directed Acyclic Graph</b>
<b>DAML</b>	<b>DARPA Agent Markup Language</b>
<b>DAML-S</b>	<b>DARPA Agent Markup Language services</b>
<b>DAML+OIL</b>	<b>Darpa's Agent Markup Language + Ontology Inference Layer</b>
<b>DHT</b>	<b>Distributed Hash Table</b>
<b>DICOM</b>	<b>Digital Imaging and Communications in Medicine</b>
<b>DIF</b>	<b>Data, Information and File resource</b>
<b>DIT</b>	<b>Directory Information Tree</b>
<b>DS</b>	<b>Data Source</b>
<b>DSR</b>	<b>Data Service Resource</b>
<b>Eclipse</b>	<b>Eclipse Integrated Java development environment</b>
<b>EGEE</b>	<b>Enabling Grids for E-scienceE</b>
<b>EII</b>	<b>Enterprise Information Integration</b>
<b>EPR</b>	<b>Electronic Patient Record</b>

<b>FSQDP</b>	<b>Full Search Query and Discovery Protocol</b>
<b>GApps</b>	<b>Grid Applications</b>
<b>GEMSS</b>	<b>Grid-enabled Medical Simulation Services</b>
<b>GIS</b>	<b>Grid Information System</b>
<b>GIIS</b>	<b>Grid Index Information Service</b>
<b>GRAM</b>	<b>Globus Resource Allocation Manager</b>
<b>GRC</b>	<b>Grid Resource Classifier</b>
<b>Grid-SD</b>	<b>Grid-Service Discovery</b>
<b>GRIP</b>	<b>Grid Interoperability Project</b>
<b>GRIS</b>	<b>Grid Resource Information Service</b>
<b>GT</b>	<b>Globus Toolkit (GT-4, GT-3, etc.)</b>
<b>GUI</b>	<b>Graphical User Interface</b>
<b>HL7</b>	<b>Health Level 7</b>
<b>HTTP</b>	<b>HyperText Transfer Protocol</b>
<b>GDS</b>	<b>Grid Data Services (OGSA-DAI service)</b>
<b>JDK</b>	<b>Java Development Kit</b>
<b>JRE</b>	<b>Java Runtime Environment</b>
<b>JSP</b>	<b>Java Server Page</b>
<b>I/O</b>	<b>Input/Output</b>
<b>ICD</b>	<b>International Classification Diseases</b>
<b>IDB</b>	<b>Incarnation Data Base</b>
<b>IK</b>	<b>Interval Keeper</b>
<b>InstruGrid</b>	<b>Instrument Grid</b>
<b>KBR</b>	<b>Knowledge Base Repository</b>
<b>KDS</b>	<b>Knowledge Discovery Service</b>
<b>KEPR</b>	<b>Knowledge Execution Plan Repository</b>
<b>KMR</b>	<b>Knowledge Metadata Repository</b>
<b>KSK</b>	<b>Keyword-Signed Key</b>
<b>LADP</b>	<b>Lightweight Directory Access Protocol</b>
<b>LAN</b>	<b>Land Area Network</b>
<b>MAGA</b>	<b>Mobile Agent based Grid Architecture</b>
<b>MDS</b>	<b>Metacomputing Directory Service (MDS-1, MDS-3, etc.)</b>



<b>MedDRA</b>	<b>Medical Dictionary for Regulatory Activities</b>
<b>MeSH</b>	<b>Medical Subject Headings</b>
<b>MIP-Grid</b>	<b>Grid-enabled Medical Image Processing Application System</b>
<b>MPI</b>	<b>Message Passing Interface</b>
<b>MOWS</b>	<b>Management of Web Services</b>
<b>MQA</b>	<b>MySQL Query Administrator</b>
<b>MRI</b>	<b>Magnetic Resonance Imaging</b>
<b>MUWS</b>	<b>Management Using Web Services</b>
<b>MySQL</b>	<b>MySQL database</b>
<b>NDMA</b>	<b>National Digital Mammography Archive</b>
<b>OASIS</b>	<b>Organization for the Advancement of Structured Information Standards</b>
<b>OGSA</b>	<b>Open Grid Services Architecture</b>
<b>OGSA-DAI</b>	<b>Open Grid Services Architecture – Data Access and Integration</b>
<b>OGSI</b>	<b>Open Grid Services Infrastructure</b>
<b>Oracle</b>	<b>Oracle Database</b>
<b>P2P</b>	<b>Peer-to-Peer networks</b>
<b>PACS</b>	<b>Picture Archiving &amp; Communication Systems</b>
<b>PDP</b>	<b>Prioritized Dissemination Protocol</b>
<b>PDDS</b>	<b>Physically Distributed Data Sources</b>
<b>PMML</b>	<b>Predictive Model Markup Language</b>
<b>PharmaGrid</b>	<b>Pharmaceutical Grid</b>
<b>PostgreSQL</b>	<b>PostgreSQL Database</b>
<b>QoS</b>	<b>Quality of Service</b>
<b>RAM</b>	<b>Random Access Memory</b>
<b>RD</b>	<b>Resource Discovery</b>
<b>RDF</b>	<b>Resource Description Framework</b>
<b>RI</b>	<b>Routing Indices</b>
<b>RIC</b>	<b>Resource Information Community</b>

<b>RMS</b>	<b>Grid Resource Management Systems</b>
<b>ROM</b>	<b>Read Only Memory</b>
<b>RS</b>	<b>Resource Space</b>
<b>RSM</b>	<b>Resource Space Model</b>
<b>RT</b>	<b>Routing Transferring</b>
<b>SD-RT</b>	<b>Shortest Distance Routing-Transferring</b>
<b>SimGrid</b>	<b>Simulation Grid</b>
<b>SNOMED</b>	<b>Systematized Nomenclature of Medicine</b>
<b>SOA</b>	<b>Service Oriented Architecture</b>
<b>SOAP</b>	<b>Simple Object Access Protocol</b>
<b>SOI</b>	<b>Service Oriented Infrastructure</b>
<b>SQE</b>	<b>Semantic Query Engine</b>
<b>SQL</b>	<b>Structured Query Language</b>
<b>SSK</b>	<b>Signed-Subspace Key</b>
<b>TCP</b>	<b>Transmission Control Protocol</b>
<b>UDDI</b>	<b>Universal Description, Discovery and Integration</b>
<b>UID</b>	<b>Unique Identification (e.g., primary key)</b>
<b>UMLS</b>	<b>Unified Medical Language System</b>
<b>URL</b>	<b>Uniform Resource Locator</b>
<b>VDG</b>	<b>Virtual Data Grid</b>
<b>VDS</b>	<b>Virtual Data Source</b>
<b>VO</b>	<b>Virtual Organization</b>
<b>W3C</b>	<b>World Wide Web Consortium</b>
<b>WS</b>	<b>Web Services</b>
<b>WSDL</b>	<b>Web Service Description Language</b>
<b>WSDM</b>	<b>Web Services Distributed Management</b>
<b>WSN</b>	<b>WS-Notification</b>
<b>WSRF</b>	<b>Web-Services Resource Framework</b>
<b>WUI</b>	<b>Web-based User Interface</b>
<b>Xindice</b>	<b>XML database system</b>
<b>XML</b>	<b>eXtensive Markup Language</b>

## Author's Declaration

This thesis gives an account of the research undertaken solely by the author during the period of November 2004 to July 2007. Some of the material contained therein has been presented as shown below:

Naseer, A. and Stergioulas, L. K. (2005). Resource Discovery in Computational Grids: State-of-the-art and Current Challenges. *Self-Organization and Autonomic Informatics (I)*, IOS Press, Amsterdam, Netherlands, pp. 237-245.

Naseer, A. and Stergioulas, L. K. (2006a). Discovering HealthGrid Services. In *Proceedings of the IEEE International Conference on Services Computing (SCC'06)*, pp.301-306.

Naseer, A. and Stergioulas, L. K. (2006b). Integrating grid and web services: A critical assessment of methods and implications to resource discovery. In *Proceedings of the 2nd Workshop on Innovations in Web Infrastructure*. Available online <http://www.wmin.ac.uk/~courtes/iwi2006/naseer.pdf>. Last accessed 27th July, 2007.

Naseer, A. and Stergioulas, L. K. (2006c). Resource discovery in grids and other distributed environments: States of the art. *Multiagent and Grid Systems - An International Journal*, 2(2): 163-182.

Naseer, A. and Stergioulas, L. K. (2006d). "A Taxonomy of HealthGrids and HealthGrid Resources". *11th International Symposium on Health Information Management Research (iSHIMR2006)*, Halifax, Canada, pp. 93-99.

Naseer, A. and Stergioulas, L. K. (2007). Combining web services and grid services: practical approaches and implications to resource discovery. In *Securing*

*web services: practical usage of standards and specifications*, eds. Periorellis, P., ISBN 1599046393 (Idea Group Inc, USA), Chapter 12.

Naseer, A. and Stergioulas, L. K. (under review). "A taxonomy of HealthGrids: Types of HealthGrids, resources and their discovery in the healthcare domain". *Journal of Biomedical Informatics*.

Naseer, A. and Stergioulas, L. K. (under review). Web-Services-based Resource Discovery Model and Service Deployment on HealthGrids. *IEEE Transactions on Information Technology in Biomedicine*.

Naseer, A. and Stergioulas, L. K. (under review). Convergence of Grids and Web Services: Current Challenges and Future Directions. *Communications of the ACM*.

# **Chapter 1**

## **Introduction**

### **1.1 Motivation and Research Challenge**

The rationale of this thesis is based on the identification that emerging Grid technologies hold the promise of a global information network that is far more powerful and uniquely distinct from the existing Internet framework in terms of ubiquitous access and sharing of geographically distributed resources. The semantic interoperability of geographically distributed and heterogeneous data resources is a critical issue that highlights the semantic heterogeneity challenge, which has not been fully addressed yet. The Open Grid Services Architecture Data Access and Integration (OGSA-DAI) is a powerful Grid technology that possesses strong features for providing interfaces to heterogeneous data sources on Grids in order to integrate them (Antonioletti et al., 2005). It can therefore be expected that OGSA-DAI can also be used for the semantic federation of data resources and address the challenge of semantic heterogeneity. One of the major bottlenecks in semantic federation is the mapping discovery. There are many ontologies and database schemas available that are too large to have manual definition of correspondences as the primary source of mapping discovery (Noy, 2004). The rationale of this thesis is that the contemporary Grid technologies are sufficient for providing effective and sustainable solutions to the problem of semantic federation of networked, heterogeneous, data resources in a systematic manner, making applications more scalable and manageable. Moreover, this could be achieved without using any of the industry-developed complex semantic mapping tools.

This research is motivated by the advancements made in the mainstream Grid technologies such as Globus Toolkit (GT4), OGSA-DAI, and their successful

implementation witnessed in other industries such as banking, finance, particle physics, biomedical, astrology, petroleum, and earth sciences, etc. It is, therefore, considered to be a logical next step to investigate how to facilitate the semantic federation of networked, heterogeneous data resources in a systematic manner, using these mainstream/contemporary Grid technologies.

The contribution of this thesis is an approach that uses contemporary Grid technologies for integrating heterogeneous data resources that have semantically different data fields (attributes). The approach is demonstrated using a prototype HealthGrid.

The proposed approach that leads to the ASIDS architecture is novel as it performs semantic matching at the data field-level or attribute-level and without using any of the complex industry-developed semantic mapping tools, which is the unique characteristic of ASIDS.

Thus the novelty, significance and usefulness of the proposed rational approach is that it provides a simple pragmatic solution to the extremely difficult and complex problem of semantic integration and interoperability of data resources in Grids.

## **1.2 Overview of the Problem Area**

The emerging Grid technologies hold out the promise of a global information channel that is far more powerful and uniquely distinct from the existing Internet framework. The Future Interconnection Environment (Zhuge, 2004a) is expected to usher in an era of intelligent interconnectivity by deploying an *Intelligent Computational Grid Infrastructure* that would ultimately lead to “globalization”, where humans, machines<sup>1</sup>, programs and processes act like *communication agents*, each playing a vital role remotely according to its own semantics and offering its dedicated services as an intelligent agent or resource.

---

<sup>1</sup> The term “machines” encapsulates all types of computer systems and attached peripheral devices.

Cross-communication among the various types of agents is needed on a large scale in order to enhance the capabilities and capacities of Intelligent Computational Grids. Such agents or resources are capable of sending and receiving requests/commands and act as independent intelligent communicators. In a Grid of computers, the resources could be networks, clusters of computers offering information related to various fields, memory space or storage capacity, CPU time, CPU cycles, computational power, data repositories, files, attached peripheral devices, sensors, software applications, or online instruments and data, all connected usually through the Internet and a middleware software layer that provides basic services for security, monitoring, resource management, and so forth (Foster and Iamnitchi, 2003).

Grids are Multi-Peer to Multi-Peer network architectures in which all the units (agents, nodes, resources, etc.) are interconnected in such a way that each unit is independent, however, none of the units is stand-alone or solitary. If given privilege, any unit can remotely access any other Grid unit and be accessed by another depending on its authorization criteria.

To access certain resources, a request must be made initially. In order to get the job completed in a consistent manner and for the successful completion of the required tasks, it is necessary to find and employ the right resource. If an inappropriate resource is being targeted and sent requests to, then the consistency of the job is very unpredictable, i.e. it can not be guaranteed that the job would be completed successfully. Therefore, it is very important that requests are made to the appropriate resource.

On a Grid, multiple resources are dispersed and scattered across different regions. Their disparate geographical locations, heterogeneous properties, distinct platforms and diverse dynamic statuses make these resources rather specialized in nature, and while offering much desired versatility, they are difficult to locate, utilize and manage. This difficulty in tracking down the right resource in vast interconnectivity environments raises the issue of Resource Discovery, i.e. the

process of gaining access to resources for successful completion of the job at hand. For this purpose, a consistent and controlled relationship among the various resources defined on a Grid is needed. As the cost and time taken to complete a job vary significantly, it is useful to monitor the job consumption rate. Successful allocation, aggregation, discovery, management, selection, sharing and utilization of autonomous, versatile and distributed resources operating under different authentication policies on a Grid are key issues that are not yet resolved.

Data is one of the types of resources available on Grids and their discovery faces such challenges. This thesis is concerned primarily with the data-type resources, as they are a prime example of high complexity and heterogeneity. Moreover, it is worth noting that Healthcare was chosen to be the exemplar application domain for this study and the proposed architecture was implemented on a HealthGrid environment (a Grid used in the context of healthcare). The HealthGrid example has been chosen as in healthcare the problem of data management and discovery is magnified due to the highly sensitive and complex nature of health-related data. The information contained on a HealthGrid has to be handled (stored, retrieved, shared) with care so as to meet the patient's confidentiality & privacy, as much as the other data security constraints. Moreover, health-related data needs to be integrated (semantically) for collaborative research in order to promote healthcare and facilitate patient's wellbeing.

This chapter explains the context of this study, and sets out its research aims and objectives, followed by an introduction to the rationale and the methods of the research approach used. The structure of the thesis is outlined at the end of this chapter.

### **1.3 Research Aims & Objectives**

The semantic interoperability of geographically distributed and heterogeneous data resources is a critical issue and Grid technologies hold the promise to address the challenge of semantic federation of data resources in a systematic manner,



making applications more scalable and manageable. Thus, in the light of the above rationale, this research addresses the following research question:

*“How to facilitate the semantic federation of heterogeneous data resources using mainstream Grid technologies?”*

By posing and testing (and ultimately verifying) the following hypothesis:

*“Existing mainstream Grid technologies are sufficient for providing effective and sustainable solutions to the problem of semantically federating networked (heterogeneous) data resources.”*

The aim of this research is to *explore the possibility of using the mainstream Grid technologies to semantically integrate heterogeneous data sources in an effective, efficient and user-friendly way*. To this end, a hypothesis on this possibility is proposed and tested. Accordingly, the research objectives are to attempt to:

1. conduct a comprehensive literature review on Grids and build a classification for taxonomies of the available resource discovery methods
2. conduct a comprehensive literature review on HealthGrids and classify the various types of HealthGrids in order to produce a taxonomy, which is our pilot study (prototype implementation) in HealthGrids
3. perform a technology analysis of the current mainstream Grid and Web technologies available and analyse if the mainstream Grid technologies can be used to address the data integration issues
4. produce/propose a suitable architecture that can potentially solve the problem
5. implement the proposed architecture on a pilot study to develop a HealthGrid-enabled application and to demonstrate the feasibility of semantic heterogeneity
6. evaluate the HealthGrid prototype implementation of the proposed architecture and demonstrate the feasibility of the proposed approach

The significance, usefulness and contribution of this research lies in proposing a simple solution to the very complex problem of semantic integration and interoperability of data resources in Grids.

The terms data sources (DS), data resources (DR), and data service resources (DSR) are used interchangeably throughout this thesis.

## **1.4 Context of the Exemplar - HealthGrids**

HealthGrids are designed and used specifically for clinical use and/or epidemiological studies, both representing areas where data integrity and platform compatibility are critical to the provision of consistent medical information to the various stakeholders of healthcare. These stakeholders include health specialists (doctors, physicians, and practitioners), medical lab technicians, pharmacists (drug developers, analyzers), surgeons, health analysts, medical equipment providers, healthcare organizations and even patients or the general public. All of them need a globally shared channel for their collaborative work on healthcare problems, and in one way or another will be positively influenced by the deployment of HealthGrids. HealthGrids are a means to deploy advanced healthcare at a personalized level by enabling virtualization of life sciences' resources globally and providing healthcare services at the patient's doorstep such as self-assessment, online-health management, etc. Many of the medical processes in healthcare IT lack consistency and adequate functionality due to the unavailability of adequate computation or storage resources required to perform the desired operation. Moreover, the geographically distributed and heterogeneous medical resources need to be integrated in a systematic manner so as to facilitate global healthcare access and services and to enhance the collaboration and sharing of information among the scientific community to perform group-wise operations on data such as dosage computation, clinical annotation service, group-based analysis & diagnosis, etc.

In HealthGrids the problem of data management and discovery is magnified due to the highly sensitive nature of health-related data. The information contained on a HealthGrid has to be handled (stored, retrieved, shared) with care so as to meet the patient's confidentiality & privacy, as much as the other data security constraints. There are a number of issues that are needed to be addressed in order to achieve successful data management & discovery: for example, encoding of medical terms, file format compatibility issues, ontologies matching issues, heterogeneity issues, data archiving & distributed image analysis issues, etc.

## **1.5 Research Approach**

This research was motivated by an initial literature review, on the basis of which, the hypothesis was formulated. Based on the conclusions drawn from the literature surveys and the related technology analysis, the architecture was proposed and then implemented on a HealthGrid (prototype). Finally, the prototype implementation was evaluated to test the implementation of the ASIDS architecture on the HealthGrid prototype and to demonstrate the feasibility of the proposed approach.

To address the issue of semantic data discovery on HealthGrids (or Grids in general), one of the best possible solutions was to develop a middleware, Grid-enabled application, that would be capable of the global integration of various health-related data reservoirs on a HealthGrid. The resulting data could be further used to perform the desired operations, such as aggregation, filtering, sorting or searching. Hence this data could be provided on the HealthGrids for scientific collaboration and sharing to facilitate better healthcare.

Through the literature surveys and the related technology analysis, it was found that the technologies which could be used to accomplish this research task included Globus Toolkit (GT4) and OGSA-DAI (Open Grid Service Architecture Data Integration and Access) with other Web Services technologies such as XML (Extensive Markup Language). The contribution of this research lies in proposing

an n-tier-to-n-tier application architecture *ASIDS (Architecture for Semantic Integration of Data Sources)* and its pilot implementation on a HealthGrid (prototype).

## 1.6 Thesis Outline

This thesis has been split into two parts:

**Part-I** this part contains Chapters 2, 3, and 4, which are the background chapters and are setting the context of this study

**Part-II** this part contains Chapters 5, 6, 7, 8, and 9, which contain the Pilot Study

**Chapter One: Introduction.** This chapter provides an introduction to this research. It presents an overview of the problem and sets out research question, aim and objectives. The importance of the research problem is highlighted and the approach adopted for carrying out the research is described. A roadmap to the whole thesis is provided at the end of this chapter.

### PART-I BACKGROUND CHAPTERS - SETTING THE SCENE

**Chapter Two: Technology Analysis.** A thorough study of the various Grid and Web technologies is presented in this chapter. Some of these technologies are to be used later on in the architecture proposed in Chapter 6. In order to fully understand the problem, to see the related technologies available to address the research question and to be able to propose a suitable solution, it became essential to conduct an analysis of these technologies. This chapter aims at describing not only the mainstream Grid and Web technologies, their purpose and interaction among them, but it also discusses the influence of Web Services on Grid technologies and analyses current trends for their convergence.

**Chapter Three: Review of Resource Discovery in Grids and other Distributed Environments.** A comprehensive review of the past and ongoing efforts to resolve the issue of resource discovery in Grids and other similar distributed environments is presented. The various resource discovery methods, techniques and approaches are discussed along with their advantages and disadvantages, and eventually recommendations are made with respect to practical implementations and directions of future research in Grid resource discovery. This chapter is the preamble of the overall literature review which will lead to the healthcare domain-specific state-of-the-art section in Chapter 4.

**Chapter Four: Taxonomy of HealthGrids - Types of HealthGrids, Resources and their Discovery in the Healthcare Domain.** Based on the complete taxonomies of Grid resources and categories of resource discovery methods mentioned in Chapter 3, this chapter presents a new taxonomy of HealthGrid types and problems associated with the discovery of heterogeneous resources in HealthGrids. The proposed taxonomies can serve as a basic platform from where further research could be launched or structured upon.

## PART-II: PILOT STUDY

**Chapter Five: Research Methodology.** This chapter explains the research methodology adopted for carrying out the research, its suitability with the study, the research design and the various phases involved.

**Chapter Six: ASIDS: The Proposed Architecture.** Based on the investigation of the problem through literature survey and the related technology analysis, this chapter proposes a design for an *Architecture for Semantic Integration of Data Sources (ASIDS)*. This architecture is later used in Chapter 7 to generate a prototype, in order to validate the hypothesis of this thesis, which can be stated as: *Existing mainstream Grid technologies are sufficient for providing effective and sustainable solutions to the problem of semantically federating networked (heterogeneous) data resources.* This chapter depicts the suggested application's

architecture, explains its different components, and discusses the variety of tools & technologies used. The functionality of the proposed architecture is demonstrated and finally conclusions are drawn.

**Chapter Seven: Implementation of the Experimental Prototype.** This chapter describes a practical implementation of the proposed ASIDS architecture on a HealthGrid (prototype) application named as the ASIDSApplication built in JAVA. This ASIDSApplication consists of three main components namely; a JSP page (called as DDQuery.jsp), a client Servlet (called as DataDiscoveryClient.java) and a Java class for semantic mapping (called as Mapping.java). The JSP page acted as the GUI interface and received queries from the users. Based on the user query, it then fetched pharmaceutical data from various data resources regardless of their geographical locations, heterogeneous formats and semantics (on field-level) and makes this data available on the HealthGrids. The data retrieved was displayed and could be further used to perform desired operations such as, scientific collaboration and group-wise or exploratory analysis, eventually promoting e-Health.

**Chapter Eight: Evaluation of Prototype.** The prototype developed and implemented in Chapter 7 was used to test the implementation of the ASIDS architecture on the HealthGrid prototype and to demonstrate the feasibility of the proposed approach with *single Grid Installation (GI)* and *multiple Grid Installations (GIs)*. This chapter discusses the experimental set-up for this evaluation and graphically presents the evaluation results. Two different experiments were conducted, (a) *Experiment-I* for testing out the system with one GI and large datasets (having semantically different data fields), and (b) *Experiment-II* for testing if the system works on adding more number of geographically distributed GIs (having semantically different data fields). For this reason both the experiments were conducted in different network setups. The elapsed time measurements were taken and results were plotted on the graphs. Results showed that the proposed semantic integration approach (ASIDS architecture) remains functional in both the experiments. Moreover, it is expected

that the architecture would still be manageable, reusable and flexible in case of even larger numbers of GIs and even increasing Data Sources just by making minor changes to the system configurations.

**Chapter Nine: Conclusions and Further Research.** This chapter provides a summary of the thesis and the conclusions of the research based on the literature reviews and the empirical work of this study. It addresses the issue of semantically integrating heterogeneous data sources and making them available on HealthGrids for scientific collaboration and group-wise analysis, eventually promoting e-Health. Moreover, avenues for future research are subsequently discussed, along with the extent to which these technologies would be applicable and adaptable to be adopted more widely.

# **PART-I**



## **Chapter 2**

### **Technology Analysis**

#### **2.1 Introduction**

A major part of this research is comprised of a thorough study of the various Grid and Web technologies. This technology analysis was conducted in order to see what do the mainstream Grid technologies offer in order to address the data integrity issues. These technologies are to be used in the architecture proposed in Chapter 6. In order to better understand the proposed architecture, it is important to have in advance knowledge of these technologies and therefore a technology analysis is conducted here.

This chapter describes the mainstream Grid and Web technologies, their purpose and interaction among them, and significantly discusses the impact of Web Services on Grid technologies and analyses trends for their convergence.

#### **2.2 Preliminary Research into Grid & Web Technologies**

This section serves to provide the required prior knowledge, along with some critical analysis, firstly, about a number of existing Grid technologies such as Open Grid Services Architecture (OGSA), Open Grid Services Infrastructure (OGSI), Web-Services Resource Framework (WSRF), Globus Toolkit version 4.0 (GT4), Open Grid Services Architecture Data Access and Integration (OGSA-DAI), and secondly, about Web Services technologies such as Extensible Markup Language (XML), Web Service Definition Language (WSDL), Universal Description, Discovery, and Integration (UDDI), Simple Object Access Protocol (SOAP), Service Oriented Infrastructure (SOI), Web Services Distributed Management (WSDM), etc.

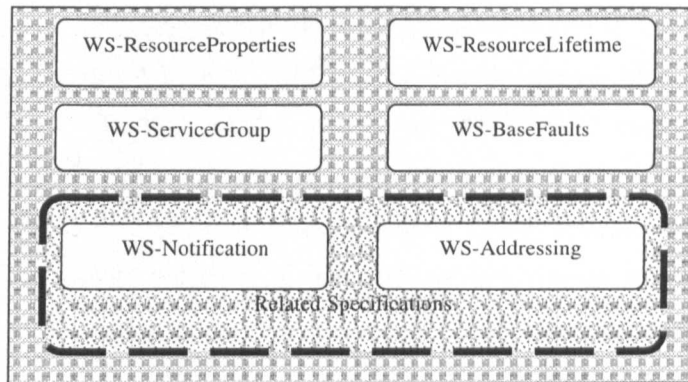
### 2.2.1 Grid Technologies (OGSA, OGSi, WSRF, GT4, OGSA-DAI)

The Open Grid Services Architecture (OGSA) defines the Grid service concept, based on principles and technologies from both the Grid computing and Web services communities (Talia, 2002). Moreover, OGSA not only defines the semantics for a Grid service, but also defines standard mechanisms for creating, naming, and discovering transient Grid service instances. It also provides location transparency and multiple protocol bindings for service instances and supports integration with underlying native platform facilities (Foster et al. 2002). Nowadays Grid services are no longer considered to be separate from the Web services. In fact, according to the Open Grid Services Infrastructure (OGSI) version 1.0 specification (Tuecke et al., 2003), a Grid service is considered to be a Web service that conforms to a set of conventions (interfaces and behaviours) which define how a client interacts with a Grid service for such purposes as service lifetime management, inspection, and notification of service state changes (Foster et al., 2005). The distributed and often sustained state of a resource is commonly required in advanced distributed applications. Such a controlled, fault-resilient, and secure management of the state is provided by these conventions, together with certain OGSI mechanisms that are associated with Grid service creation and discovery. Recently there has been a drift from OGSI to the Web-Services Resource Framework (WSRF) due to potential performance advantage reasons (Czajkowski et al., 2004a).

The Web-Services Resource Framework (WSRF) (Czajkowski et al., 2004b) is concerned primarily with the creation, addressing, inspection, and lifetime management of state-enabled resources. It codifies the relationship between Web services and state-enabled resources in terms of the implied resource pattern, which is a set of conventions on Web services technologies. A state-enabled resource that participates in the invoked resource pattern is termed a WS-resource.

WSRF has five specifications (Figure 1), namely: WS-ResourceProperties, WS-ResourceLifetime, WS-ServiceGroup, WS-BaseFaults and other related

specifications such as WS-Notification and WS-Addressing. The WSRF specifications relate to the management of stateful Web Services.

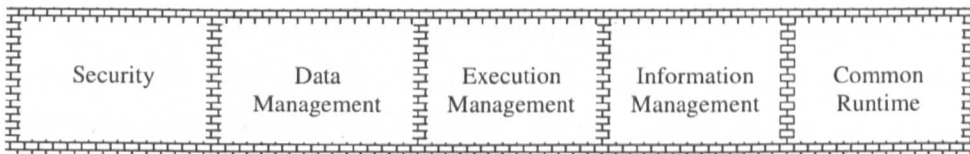


**Figure 1: WSRF Specifications**

WS-ResourceProperties is composed of the resource or service attributes, WS-ResourceLifetime is responsible for the lifecycle management of the resources or services, WS-ServiceGroup facilitates group operation of services, and WS-BaseFaults deals with the reporting of all faults occurring during the invocation of a Web Service. Other related collection of specifications, which is not part of WSRF but is quite related to it, includes WS-Notification which is responsible for notifying changes in a Web Service and WS-Addressing, which is used to address a Web Service (Sotomayor, 2007). In WSRF, Web Services are considered as resources which are stateful Web Services, also known as WS-Resources.

The WSRF framework describes the WS-resource definition and its association with the description of a Web service interface and describes how to make the properties of a WS-resource accessible through a Web service interface and to manage a WS-resource's lifetime. Based on industry feedback, the revised and updated WSRF specifications were submitted to two new OASIS technical committees, the WS-Resource Framework (WSRF) TC and the WS-Notification (WSN) TC (Baker et al., 2005). WSRF was an important step forward for the Grid community.

The Globus Toolkit (GT4) (Sotomayor, 2007) is originally based on the Web Services. GT4-Globus Container a set of Grid services that, in addition to the core Grid services, also can contains user-defined services. For instance, the user-defined Open Grid Services Architecture Data Access and Integration (OGSA-DAI) service (Karasavvas et al., 2005) runs as a customized data service in the Globus container. The latest version of GT4 uses the WSRF framework. Although the WSRF infrastructure is only a part of GT4, most of GT4 architecture is built on top of it. There has been few evolutionary transformations in the GT4 architecture (Sotomayor, 2007); notably, the non-WS version of Monitoring and Discovery Service (MDS2) has been deprecated and shall be dropped from the future releases, to be replaced by a new Web-based WebMDS component in the GT4 architecture. There are many other non-WS components in the GT4 architecture which are gradually being replaced by respective WS-based components. The future releases of Globus Toolkit are expected to be based on Web Services specifications and those components that are not Web-based will be deprecated. The Web Services implementation of Globus components has been optimized for flexibility, stability and scalability.



**Figure 2: Main Categorizations of Globus Toolkit (GT4) Architecture**

The GT4 architecture has five main categories (Figure 2), namely Security, Data Management, Execution Management, Information Management and Common Runtime. Each of these is further sub-categorized into smaller components. Currently, this architecture contains both the old (non-WS based) and the new (WS-based) components.

Open Grid Services Architecture Data Access and Integration (OGSA-DAI) (Antonioletti et al., 2005) is a powerful technology that possesses strong features for providing interfaces to heterogeneous data sources on Grids in order to

integrate them. OGSA-DAI acts as an interface for each data source that is available on the Grids. It is an open-source middleware designed to facilitate controlled access, management and integration of distributed heterogeneous data resources and provides a ready-made framework that promotes locality and product transparency to connect data resources to the Grid environment (Crompton et al., 2006).

### **2.2.2 Web Services Technologies (XML, WSDL, UDDI, SOAP, SOL, WSDM)**

Web services can communicate with other Web services regardless of their implementation method in order to make them interoperable. The current de facto standards are the following:

- Web Service Description Language (WSDL) - for describing Web Services
- Universal Description, Discovery and Integration (UDDI) - for publishing Web Services and for service registry
- Simple Object Access Protocol (SOAP) - for invoking Web Services

Web Services provide standardization, while they are both platform & language independent and most of them use HyperText Transfer Protocol (HTTP) for transmitting service requests and responses. They are self-describing and can act as stand-alone, self-contained agents, each performing dedicated tasks. For example, the service invocation process is supported by Simple Object Access Protocol (SOAP) messages, specifying a standard format of service request and service response. Similarly, HTTP is responsible for the transmission of these messages between client and server.

Web Services are self-describing, as to the operations they support and the way to invoke them. This is handled by the Web Services Description Language (WSDL), which is a special eXtensible Markup Language (XML) language, used to define a Web Service interface and specify operations that a Web Service

offers. The standard Simple Object Access Protocol (SOAP) was developed to enable the transmission and reception of messages for accessing distributed resources or objects. Moreover, the goal of Universal Description, Discovery and Integration (UDDI) specification was to provide a centralised registry for the web services by making it easier to locate them (Twardoch, 2003).

Service Oriented Infrastructure (SOI), which is about using shared services, is considered to be a combination of two technologies, namely Web Services (WS) and Service Oriented Architecture (SOA) (Globus Consortium Journal, 2006). Web Services are rapidly being implemented in combination with SOA.

The SOA paradigm has attracted not only the Web community, but also the Grid community, on the basis that it can provide a framework whereby a great number of services can be dynamically located, balanced, and managed, so that applications are always guaranteed to be securely executed, according to the principles of on-demand computing (Congiusta et al., 2007).

Web Services Distributed Management (WSDM V1.1) got approved as an OASIS standard by the OASIS Technical Committee in August 01, 2006 WSDM (Web Services Distributed Management, 2006). WSDM has two sets of specifications namely: Management Using Web Services (MUWS) and Management of Web Services (MOWS). It was developed on a set of architectural foundations, namely the Web Services architecture and the Service Oriented Architectures (SOA). The WSDM standard specifies how the manageability of a resource is made available to respective consumers via Web services. It can provide a solid, standards-based framework for managing computing resources across the IT environment or interconnected consumer devices around the globe. WSDM builds upon standards, rather than redefining or re-inventing technologies that already have strong industry footings.

Although such languages provide the technical means for achieving cross-platform distributed software deployment, they are not sufficient to achieve the

required level of semantic expression (David et al., 2005). Moreover, the statelessness of Web Services is another hurdle in the implementation of Web Services on cross-platform heterogeneous and scalable systems, which could easily be addressed by structured implementation of the Grids technology. Grids will need to seek application-oriented solutions and address WS-based real use cases (Globus Consortium Journal, 2006).

## 2.3 Convergence of Grid and Web Services Technologies

Each of the Grid and Web services are specialized so as to provide sophisticated functionalities in their own domains. Web services can not be directly implemented on Grid architecture due to constraints such as their stateless nature and persistency; whereas the Grid services should always have a state and are transient in nature. Therefore, the integration of Grid services with Web services is necessary in order to support a fully- or partially-Gridified Web (Naseer and Stergioulas, 2006b). A partially-Gridified Web has an infrastructure network that is based on the Web but has a number of small Grid networks (or Grid infrastructures) built on top of it. It would provide a platform for carrying out distributed execution and remote processing of any dataset through intensive computation, for storing huge masses of data and for supporting group-wise collaborative analysis. Where group-wise collaborative analysis includes remotely analyzing a particular problem in groups or communities (online) such as group of pharmacists analyzing the effects of a drug or a group of scientists carrying out disease analysis through performing an online simulation of the body organ. Users of the partially-Gridified Web resources can be human users, such as computer operators, system administrators, programmers, scientists, or some automated processes or computer programs that send commands/requests to use or discover a Grid resource. Web Services are the technology of choice for Internet-based applications with loosely coupled clients and servers, which makes them the natural choice for building the next generation of Grid-based applications. However, Web Services do have certain limitations. In fact, plain Web Services (as currently specified by the W3C) would not be very helpful in building a Grid

application. The gap is filled by WSRF, which promises to improve several aspects of Web Services to make them more adequate for Grid applications. For instance, Web services can not be directly implemented on Grid architecture, due to constraints such as their stateless nature and persistency; whereas the Grid services should always have a state and are transient in nature. In WSRF, a separate entity called Resource stores all the state information; it contains meta-data about a Web Service (Sotomayor, 2007).

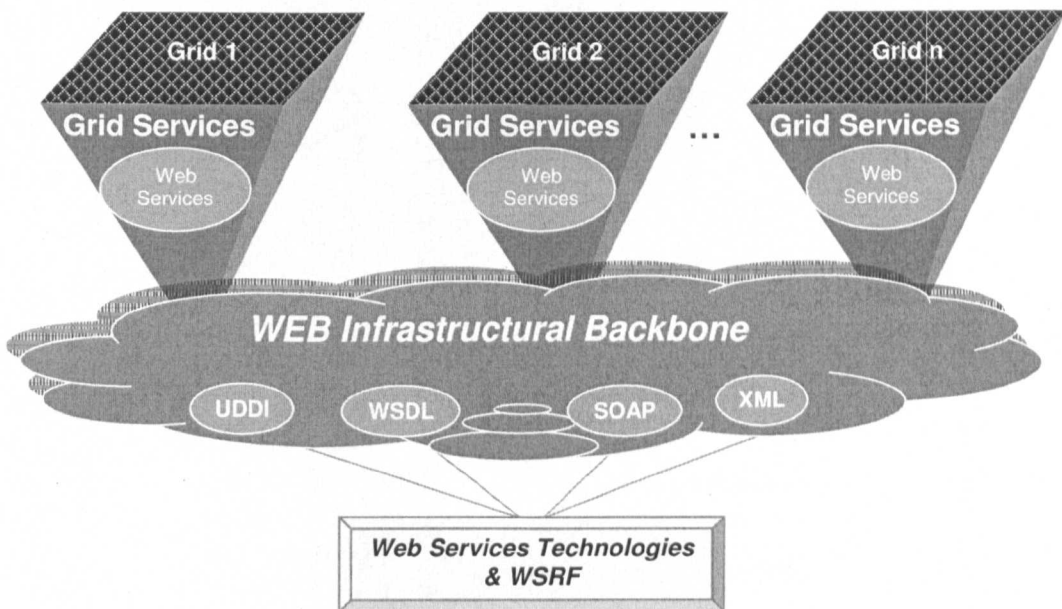
Web Services are a way of not only encapsulating application functionality in both implementation- and location-transparent manner, but also of packaging features and making them accessible to other businesses as distributed software components. However, rapid changes in the Web Services, (e.g. introduction of new Web Services into a dynamic business environment) can lead to undesirable results and poor service quality. Web Services may interact with each other in unexpected and undesirable ways and lead to undesirable interactions (Weiss et al., 2007).

It can be clearly seen that both the Grids and Web Services have individual loop holes and drawbacks that lead to many potential technical problems. However, their convergence could help achieve their integration in a mutually complementary manner. This convergence can fill the gap in successful implementation of Grids, at very least by providing a standard *interface* - a Web Services-based interface for Grid applications. Easy and user-friendly interfaces from Web Services combined with complex Grid Technology at the back-end can make this convergence a mutually beneficial relationship. The idea of integrating Grids and Web Services, together with some possible scenarios for this integration, has been presented (Naseer and Stergioulas, 2006b). In particular, two different approaches to convergence have been discussed. The first approach, called Grid-based Web services, suggests building new Web Services that are based on Grid standard interfaces and behaviours (inherited or encapsulated) to make them operable on the Grids. Whereas the second approach, known as Web-based Grid services, suggests building new individual Grid services that are based



on Web Services standards and contain the features & functionalities of Web services (inherited or extended). Either of these integration approaches could be a good candidate for a viable solution to the Grids and Web Services convergence problem, according to their respective implementation environments.

However, due to the vital differences in the standardization status of Grids and Web Services and the lack of standard interfaces in Grids, the second approach is more appropriate and recommended for implementation as shown in Figure 3, (Naseer and Stergioulas, 2007).

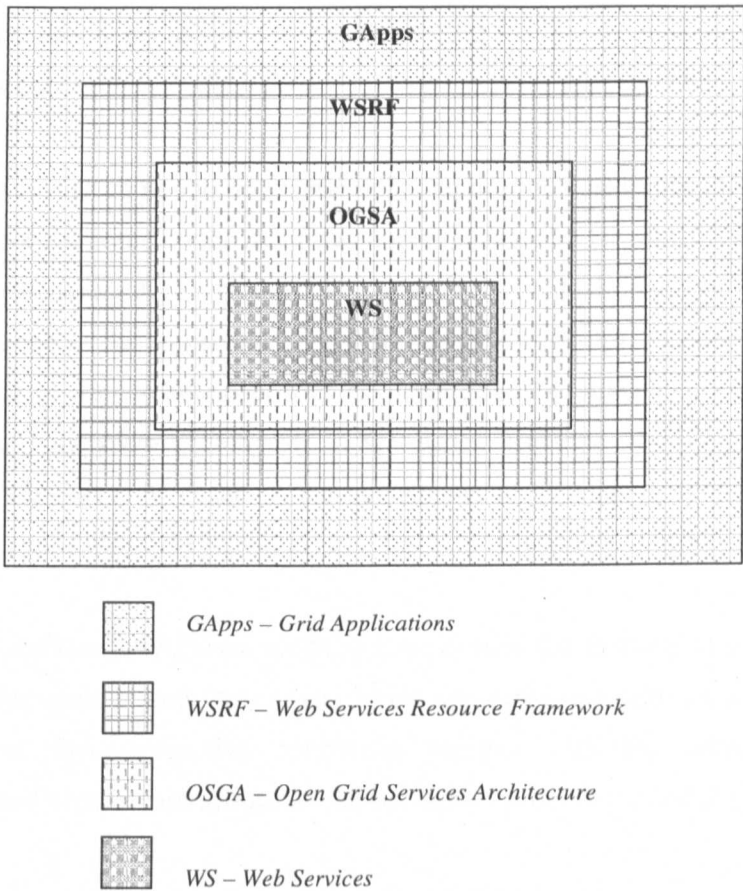


**Figure 3: Web-based Grid Infrastructure**

Moreover, the integration of applications and tools which are often incompatible, for the purposes of data acquisition, registration, storage, provenance, organization, analysis and presentation, requires the use of both Web and Grid services (Sloot et al., 2006).

Web Services can support this realization by making the complexity of underlying Grid technology transparent to its users. This complexity presents a major

obstacle to the universal implementation and deployment of the Grids. The designing and deployment of a Grid infrastructure is the subject of a relatively new area, which has not yet reached a maturity stage of effective standardization.



**Figure 4: Implementation Architecture for Grid Applications**

Many of the problems associated with the implementation of Grids could be resolved if the Web is used as the underlying (“backbone”) infrastructure for the development of Grids, thus resulting in the formation of a Grid-enabled or a partially Gridified-Web. Hence, there is a need for the two technologies to be glued together using some specialized (object-oriented) techniques.

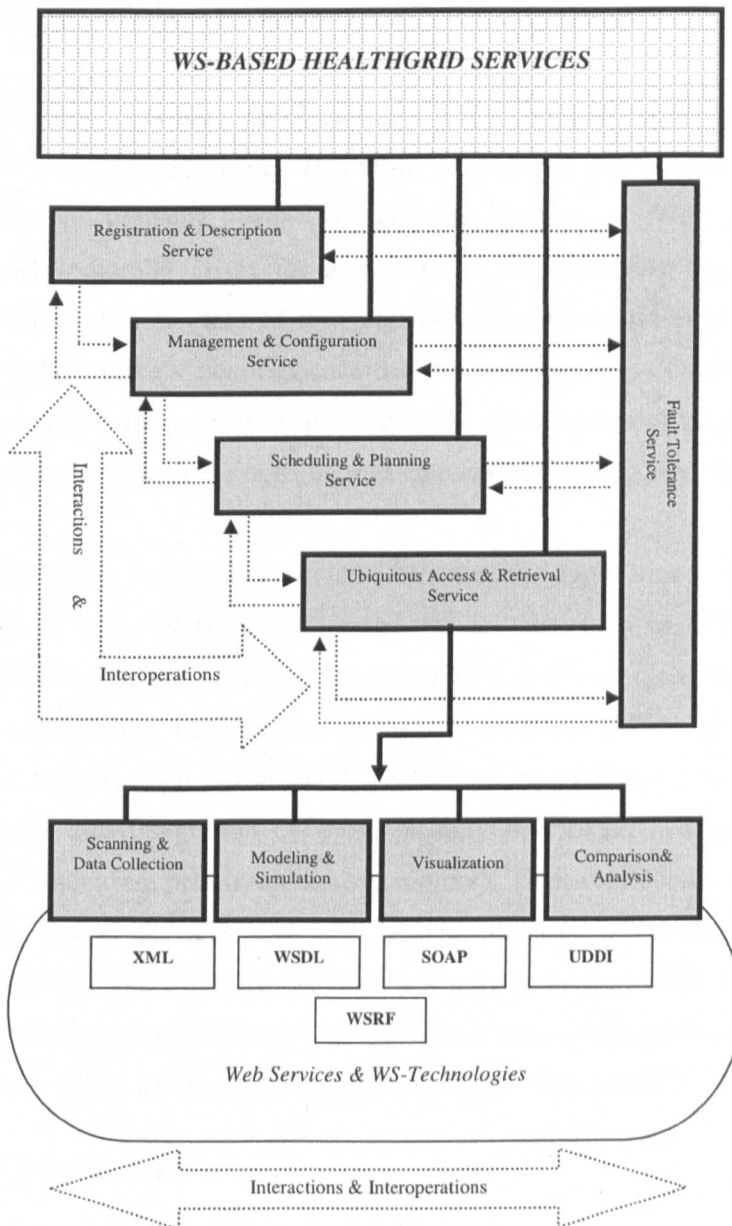
An important step towards achieving this convergence is the development of Web Services-based Grid applications. Application designs based on the Web-based Grid approach (Naseer and Stergioulas, 2007) are expected to be built on a special architecture for Grid Applications (Figure 4). The core of this architecture is composed of stateful resources within the Web Services Resource Framework (WSRF) which is built using Web Services (WS), upon which Grid Applications are built and which are required by the Open Grid Service Architecture (OGSA) standardisation.

As Healthcare was chosen to be our exemplar application domain for this research study and our proposed architecture was implemented on a HealthGrid prototype, for the verification of our hypothesis. It has been seen through the literature review that one of the biggest challenges in HealthGrids is the integration, access and retrieval of data in heterogeneous environments, also maintaining quality of service (QoS) and security alongside. A WS-Based Resource Discovery Model is presented (see Figure 5) based on the cornerstone of developing Web Services-based Grid applications for providing services in the healthcare sector.

The WS-based Resource Discovery Model shows how the HealthGrid services at both the Management and Operational levels are interacting with each other. A hierarchy of their respective occurrence together with the intra-services, horizontal and vertical interactions & interoperations is demonstrated in Figure 5.

This model is based on Web Services and the WS-Technologies that are contained or encapsulated into the HealthGrid services at both the Operational and Management levels (Naseer and Stergioulas, 2006a). The HealthGrid services infrastructure presented in this model supports the theme of horizontally coupled or integrated and vertically decoupled or disintegrated technologies and standards (Foster and Tuecke, 2005). The Web Services Technologies and HealthGrid services are integrated or converged horizontally to facilitate or provide a combined functionality. This model can be used as the basis for designing an

infrastructure/architecture of Grid applications (as proposed in Chapter 6, which could be applied to a HealthGrid context as well (as implemented in Chapter 7).



**Figure 5: WS-Based Resource Discovery Model: HealthGrid Example** (the low level services shown in the model are taken from the HealthGrid exemplar)

For the e-Health dream to become reality, successful implementation of these concepts is crucial. Moreover, it is necessary for web services that enable remote access to medical data to be secured in an appropriate fashion (Power et al., 2006). Web services could be wrapped using generic SOAP proxies.

Grid technology has been developed as a way to support large-scale distributed resource manipulation, using shared heterogeneous pooled resources across administrative domains. With the growing success of Web Services, the opportunity naturally arises for a confluence between these two increasingly mature areas. A new class of distributed applications could even emerge as a result - in fact, there's clear evidence that this is happening. Certain applications, called pervasive management support systems, combine sensor network and Grid technologies to support the ongoing management of evolving physical systems.

Grids will need a new orientation towards enterprise applications, to replicate the successes of other Web Services-based technologies such as WSRF, SOI and WSDM. However, the Grid applications have to deal with many problems, such as capacity, quality, network management or scalability. The main drivers for Grids are features such as the speed and level of granularity e.g. in data analysis applications and integration of geographically distributed heterogeneous Grid resources (including peta-bytes of data sources). Their role should be to provide not only shared services, but also shared physical resources to fulfil the community demand. Effective use of the above features must be more emphasized in Grid deployment for domain-specific and real-world problem solving in enterprises, within a converged infrastructure using Web Services.

## **2.4 Conclusion**

The aim of this chapter was to describe the mainstream Grid and Web technologies, their purpose and interaction among them and to see whether any of these could be used for resolving the data integrity issues. Moreover, the influence of Web Services on Grids was discussed and trends for their confluence are

analysed. It has been seen that few of the mainstream Grid technologies, mainly GT4 and OGSA-DAI, have been explored and can provide candidate solutions to the semantic data integrity issue and this paves our way to the proposed architecture.

## Chapter 3

# Investigation of Resource Discovery Methods in Grids and other Distributed Environments

### 3.1 Introduction

The emerging Grid technologies hold out the promise of a global information channel that is far more powerful and uniquely distinct from the existing internet framework (Naseer and Stergioulas, 2006c). Grid technologies could be used to facilitate many activities such as sharing of geographically distributed autonomous resources, collaborative or exploratory analysis of large datasets to conduct experiments, clinical trials for drug discovery or enable research collaboration within virtual organizations, etc. Seamless and loose integration of diverse and heterogeneous resources is an essential part of the resource discovery process. In order to integrate heterogeneous information, it is important to first have controlled access to the available data and information resources.

Although a large body of literature is now accumulated on the area of Grid resource discovery, the diversity of the problems and the range of applied methods make it difficult to understand the sometimes subtle problems of Grid resource discovery in a well-structured and systematic manner. Thus, a detailed review would be quite beneficial so as to provide background knowledge of the past and ongoing efforts for achieving Grid resource discovery, to the wider research community, not only the (new or experienced) researchers in this field, but also researchers from other fields that would like to familiarize themselves with the subject of Grid technology. Moreover, the proposed taxonomy would serve as a basic platform from where further research could be launched or structured upon. Thus, this chapter presents a detailed review of the current state-of-the-art of the various resource discovery methods available and provides an up-to-date

taxonomy of existing Grid resource discovery methods. The chapter begins with an introduction to the Grid technology, then categorizes different types of resources available on Grids, and discusses the mechanism of resource discovery and the need for discovering resources, highlighting the issues and technical limitations. Moreover, a taxonomy of various resource discovery methods is presented in terms of three different models. These models and various resource discovery approaches are then critically discussed, potential problems are highlighted and recommendations are made with respect to the practical implementation and directions for future research in Grid resource discovery.

The chapter explores the resource discovery challenges for all types of Grid resources such as storage, computation, etc. It should be noted that the work of this thesis is focused on data / and information / type resources on HealthGrids, due to the high complexity and large volumes of data available them. HealthGrids were chosen to be our exemplar application domain for this study.

## 3.2 Introduction to Grids

A Grid is a type of parallel and distributed system that enables the sharing, selection and aggregation of geographically distributed ‘autonomous’ resources dynamically at run time depending on their availability, capability, performance, cost, policies or rules which govern them and the user’s quality-of-service requirements. This sharing is, necessarily, highly controlled, with resource providers and consumers defining clearly and comprehensively what exactly is to be shared, who is allowed to partake in this “sharing”, and the conditions for sharing. A set of individuals and/or institutions defined by such sharing rules is called a *virtual organization* (VO) (Foster et al., 2001).

The concept of Grids emerged in late 1990’s when Ian Foster (Foster and Kesselman, 1999) introduced the Grid as: “A Grid is a hardware and software infrastructure that provides dependable, consistent, pervasive, and inexpensive access to high-end computational capabilities”.



Since then, many definitions and terminologies have been given to this relatively new concept. In a broader view, Grid can be viewed as a paradigm, an emerging technology that aims to change the perspective of not only today's computer usage but also of computer resources and computer networks. Based on the literature study about the Grids, it is attempted to offer an updated and comprehensive definition of a Grid:

*"Grid is a large-scale, high-performance, always-on and dynamic, although geographically distributed yet networked, infrastructure that comprises and seamlessly unifies a variety of autonomous, heterogeneous components such as processes, resources, network layers, interfaces, protocols and services, with strong, consistent and controlled relationships among them."*

It would not be far off the mark to describe the Grid as a vast network, but calling it merely a computational network will not do it justice in the semantic sense; it would rather be more fitting to view a computer network or a cluster of computers as one of the Grid resources (if it is registered on a Grid). The Grid infrastructure, being distributed in nature, allows for high variability in user and resource participation. It deploys a decentralised computing environment, which is compatible or interoperable with all sorts of network architectures.

In spite of being heterogeneous and distributed in terms of resources, a Grid system differs quite significantly from the conventional distributed systems and resource sharing environments, such as P2P networks and clusters. It provides abstraction at both the user and resource level, which is transparent to the user and relies on a standards-based service infrastructure to share computers, storage space, sensors, software applications and data, etc. across organizational boundaries. Grids provide a platform to support various (distributed) applications via resource sharing.

Compared to Grid environments, P2P systems provide limited, specialized functionality to larger and less homogeneous communities (Iamnitchi and Foster,

2004). P2P systems are potentially unreliable, whereas Grids are much more reliable and ideal to cater for professional organizations.

In clusters, resource management is performed by a centralised resource manager and the nodes cooperatively work together as a single unified resource; whereas in Grids, each node has its own resource manager and the aim is not to provide/support a single system view. Thus in Grids autonomous resources are managed by distributed resource managers. Each and every node on a Grid is considered to act both as client and server simultaneously.

### **3.3 Types of Resources on Grids**

A resource can be any real or conceptual object that is needed to be accessed by other entities, such as human users of the system or programmes that generate requests for accessing particular resources.

The types of resources available on a Grid are generally more powerful, more diverse, and better connected than the typical P2P resource. A Grid resource might be a cluster, storage system, database, or scientific instrument of considerable processing/computation value that is administered in an organized fashion according to some well-defined policy (Foster and Iamnitchi, 2003).

Figure 6 depicts the categorization of Grid resources in a simple hierarchical model where the two major categories are Physical and Logical resources.

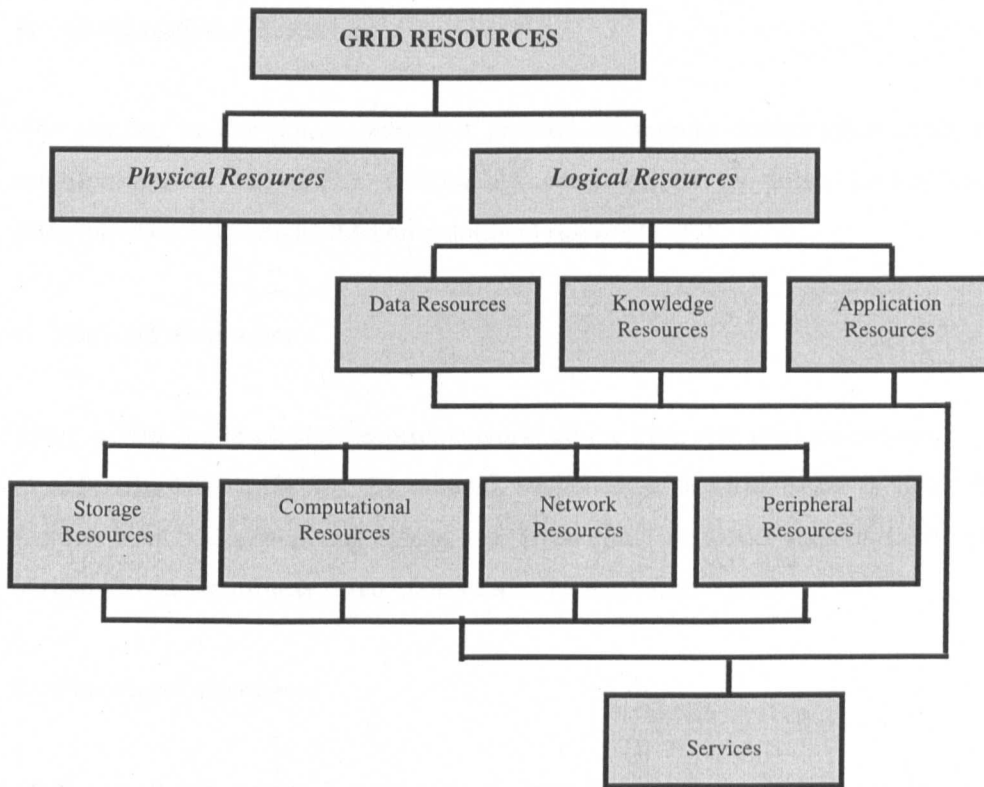


Figure 6: Hierarchy of Grid Resources

### 3.3.1 Physical Resources

Physical resources on a Grid are the tangible, hardware components that build up its infrastructure. In order to meet a demand or request for resource discovery, these can be used as a means to also obtain logical resources on a Grid. Four major sub-categories of physical resources are storage, computational, network and peripheral resources:

#### a. Storage Resources

All the storage devices (primary, secondary, internal, external, etc.) operating on a Grid come under the category of storage resources. For example, Hard Disks, RAM and ROM memories, Disk Drives, Buffer devices, etc.

***b. Computational Resources***

The devices and/or components that provide or support computation, such as microprocessors and CPUs (commonly characterized by time, cycles, and throughput) are known as the computational resources of the Grid.

***c. Network Resources***

Since a Grid is a large-architecture network, all the hardware devices encountered in networks also make up the network resources of a Grid, such as network routers, hubs and connecting cables, etc. Even small networks such as LANs or Virtual Organizations (as single units) can come under this category.

***d. Peripheral Resources***

All the input and output devices such as printers, scanners and scientific devices such as particle accelerators, Magnetic Resonance Imaging (MRI) machines, telescopes, etc., are termed as the peripheral resources on a Grid. A user on the Grid might request access to a particular peripheral device for carrying out specific I/O tasks.

**3.3.2 Logical Resources**

Unlike physical resources, the logical resources are non-tangible and constitute the driving force of a Grid. They support the Grid's hardware operations and, at times, their discovery task is requested in an indirect way. The logical resources are further sub-categorized into data, knowledge and application resources:

***a. Data or Information Resources***

All the facts and figures related to a specific domain and/or organization (particularly VOs) are considered to make up the data resources of a Grid. This data could be used to retrieve useful information and therefore authorized users

might need to access a particular organization's data to extract some information. Files also come naturally under the category of data or information resources, therefore the various file sharing and/or discovery systems are considered as resource discovery systems on a Grid.

#### ***b. Knowledge Resources***

Knowledge resources are emerging as one of the most important type of resources on a Grid. All the other types of resources, such as data, storage, etc. are accessed and used with the ultimate aim to extract some sort of specific knowledge from them that could be of utmost importance to various users, belonging to different communities or VOs, who access the Grid and place requests for acquiring knowledge (as a result of intelligent resource discovery).

#### ***c. Application Resources***

Various applications and software programs that are installed on Grid hardware fall under this category. Since running an application requires some form of hardware, the application resources are always executed and accessed by other Grid resources. All application programs, and even computer operating systems, are considered to be as application resources on a Grid. Applications always demand some sort of change from the system, which is one of the reasons for making Grid a dynamic infrastructure since the number, status and types of resources offered is ever changing (dynamic).

#### ***d. Services***

Services are used to access other types of Grid resources. All the Grid resources mentioned above are utilized and accessed to get some sort of specific services. This service could be in the form of data manipulation (storage/retrieval) using a storage resource, getting solutions to complex problems, calculations via computational resources, or accessing and using a hardware device, such as a

printer, etc. A Grid information service must provide information about all Grid resources (including services), and should minimize the number of persistent information servers that have to be managed in order to enable Grid services and applications (Johnson and Brooke, 2002).

Service providers are entities that provide access and retrieval of resources to various users. Service providers are responsible for providing accurate information about a particular resource (usually in the form of metadata for the service they are providing). They contribute to essential and authentic resource discovery.

Here services are treated as a type of Grid resources that are used to access other services and hence other types of Grid resources. The service-type resources can be categorized into the Operational-Level and Management-Level services (Naseer and Stergioulas, 2006a). The former are responsible for performing all types of tasks/operations that would fulfil a Grid user's query or request, whereas the latter are responsible for the management of Operational-Level services and for making them available on the Grids. To access services, one would need to discover them first. Thus a service has to be discovered first, before it is triggered to further access another resource. In this way, the discovery process becomes a recurrent, cyclic process, in which Grid services can be used to access other services, i.e. one type of resource is used to access other types of resources.

Both the physical and logical resources are strongly integrated and support each other in carrying out the various operations on a Grid. Each of these resources has individual availability status and is responsible for performing some specific tasks. Resources are located in geographical regions belonging to different time-zones and are heterogeneous - as their properties, capabilities, configurations and status change over time. Moreover, resource attributes, being dynamic in nature, need to be updated periodically.

Users of Grid resources can be human users, such as computer operators, system administrators, programmers, business or enterprise users or some automated processes or computer programs that send commands and/or requests to use or discover a specific Grid resource.

### **3.4 Resource Discovery on Grids**

On a Grid, various resources are dispersed or scattered in different regions. Their heterogeneous geographical locations, different platforms and dynamic status make these resources versatile in nature and therefore difficult to manage. It is not easy to track or locate the right resource in a vast interconnected environment. The mechanism of resource discovery can be viewed through different lenses in various domains; it is a multi-disciplinary task and is one of the most important issues to be dealt with in the future Grid technology. For the successful deployment of a Grid infrastructure, it is essential to access and make maximum use of the resources that are available on the Grids and this is only possible if the resources are tracked effectively and efficiently. Although resource discovery is quite a familiar term, however due to the rapid advancements in technology, an updated and comprehensive definition of Resource Discovery is presented here as follows:

*“Resource Discovery is the operation of tracking, accessing, matching, selecting and eventually requesting the right or the most accurately suitable resource for the successful accomplishment of the desired job”.*

For each job, the expected cost and time consumed by the job has to be taken into consideration and be monitored. Each resource has to comply with standard connectivity protocols for communication and security, and other resource-specific protocols for enquiry, allocation, and management (Foster et al., 2001).

Techniques adopted for resource discovery should be both location and platform independent. When a request is placed for some particular resource, the entire

network is first searched to track or locate a suitable resource, and then the resource is matched against the request query and selected (if a sufficient match is established). Upon selection its availability status is checked and, if available, the desired task is performed, hence completing the resource discovery process.

### **3.4.1 The Need for Resource Discovery**

In the face of recent technological advancements in different fields of computer science, there is a need for an infrastructure that would assist society to cope with and make maximum use of these rapid advancements. The infrastructure known as Grid promises to fulfill these expectations and provides a platform for carrying out remote processing through intensive communicational ability, providing petabytes of storage and facilitating seamless resource access & control for integration. For successful deployment of a Grid model, it is necessary to discover the right resources available on it. Allocation of the appropriate resources is difficult in a Grid environment since Grid resources vary in many aspects such as diversity, geographical distribution, large volumes and their dynamic behaviour. Due to these issues the discovery, characterization, sharing and monitoring of resources are challenging problems for the Grid community.

The challenging issues for on-demand applications derive primarily from the dynamic nature of resource requirements and the potentially large populations of users and resources. These issues include resource location, scheduling, code management, configuration, fault tolerance, security, and payment mechanisms (Foster and Kesselman, 1999).

Moreover, today's applications demand uniform access and control to huge volumes of data, held in different types of distributed data resources, that could be used for distributed data analysis (of large datasets). Therefore, effective integration of heterogeneous data resources have become a key requirement to make feasible large collaborative environments (Karasavvas et al. 2005). Complex data retrieval queries require access to data and information resources to integrate



the heterogeneous information and this involves data collection from various geographically distributed data stores to obtain the latest data on all relevant variables and making all this transparent to the end-user (Jeffery, 2007). Resource discovery is really an enabler for bringing idle system resources to use and would facilitate the seamless integration of structured, heterogeneous data resources and making them federated over the Grid for dynamic information retrieval.

### **3.4.2 Grid Resource Discovery Issues & Technical Limitations**

Issues like geographical dispersion, heterogeneity, large number of users (requesting for a particular Grid resource), dynamic nature and status of resources make resource discovery a challenge in the deployment of Grid systems. Resources of the same type can be highly heterogeneous (Iamnitchi and Foster, 2004), such as computers with different operating systems, number of CPUs and speed, datasets of various types and sizes, services of various sorts, etc. Heterogeneity in itself is not a small issue; it encompasses all aspects of compatibility conflicts, such as differences in operating systems, platforms, domains and protocols, etc. Moreover, achieving efficient job execution in a Grid environment constrained by deadlines and budget constraints is a complicated task (Chapman et al., 2001).

The need for resource discovery mechanisms on Grids emerged from technical limitations such as:

- Autonomous, heterogeneous resources
- Dynamic nature & status of resources
- Geographical dispersion of resources
- Large volumes of data/information
- Large number of users and large distributed networks
- Different operating systems/platforms
- Different administrative domains
- Lack of portability

- Availability status of resources
- Different technology policies

To resolve all these issues, Grids need a consistent, efficient, time-saving and cost-effective resource discovery mechanism. To make the discovery of resources more powerful, various resource discovery techniques and approaches have been proposed and applied to different settings, but no generic/comprehensive solutions to this problem have yet emerged. Moreover, a representative taxonomy of resource descriptions might prove to be a powerful tool for engineering resource discovery solutions. What still needed is powerful, platform-independent and multi-user handling resource discovery architecture to support Grid environments.

### **3.4.3 Expected Benefits from Resource Discovery in Grids**

If effective solutions to the issues of resource discovery are provided then the expected benefits may include:

- Efficient resource allocation
- Optimal distribution of “Grid Power”
- Maximization of usage of resources
- Increased usefulness/impact of Grid technology
- Success in deploying an infrastructure more powerful than the Internet

From the above, there is a clear need for a powerful, platform independent and multi-user handling resource discovery architecture to support Grid environments.

## **3.5 Taxonomy of Resource Discovery Methods**

To resolve the issue of Resource Discovery in Grids, different methods have been devised. Three alternative models were proposed (Buyya et al., 2000b) for modelling the Grid Resource Management Architecture; namely, the *hierarchical model* that represents the approach followed in many contemporary Grid systems, the *abstract owner model* that follows an order and delivery approach in job

submission and result gathering and the (*computational*) *market model* that brings together the essentials of both hierarchical and abstract owner models and uses the concept of computational economy in the development of Grid resource management systems.

Another study proposes a Taxonomy of Grid Resource Management Systems (RMS) to capture the essential components and functions of a Grid RMS (Krauter et al., 2002). Various resource discovery/management systems and architectures are compared to study the architectural approaches used and issues that are unresolved. The taxonomy has been developed by using the described requirements for RMSs and a developed abstract functional model. The taxonomy has focused on the type of Grid system, machine organization, resource model characterization, and scheduling characterization. It is assumed that the RMS operates on a 'globally' named pool of resources. Several current generation RMSs include "naming" as an internal function of the RMS. Further research is necessary to closely examine the trade-offs of having a naming function. One motivation for making "naming" a global function is that it facilitates interoperability between different RMSs, which may be essential for the Grid to scale to Internet proportions.

Another approach (Zhuge, 2004b) employs a Resource Space Model (RSM), which uniformly specifies and organizes resources in normal forms by using a hierarchy of top-down partitioning and a Resource Space (RS), which is effectively a semantic coordinate system with independent coordinates and mutually-orthogonal axes. The design method integrates assistant tools, an experience-based design process and strategy, and the RSM reference model. It proposes a four-step method for designing the logical-level resource spaces namely, *resource analysis*, *top-down resource partition*, *design of two-dimensional resource spaces*, and *joining of resource spaces*. Results show that (a) it is possible to transform a relational table to a resource space, (b) the transformation can keep the normal form correspondence between the relational model and the RSM, (c) the 3-D resource space can manage multiple relational

tables and (d) the application scope of the RSM is wider than that of the relational data model.

An analysis of existing resource configurations, is presented in (Yang-Suk et al., 2004), which proposes a Grid platform generator that synthesizes realistic configurations of both computing and communication resources. It proposes the development of models for resources available in current Grids, extrapolating from these models to systems.

This section examines the various resource discovery approaches based on three architectural models (centralised, distributed and semi-distributed) adopted so far for developing resource discovery solutions. Here a taxonomy of resource discovery methods is proposed in terms of three architectural models. Figure 7 depicts a taxonomy of resource discovery methods in Grids, where the main models - Centralised, Distributed and Semi-Distributed - are further sub-categorized. The three models and their applications along with their strengths and weaknesses are compared and discussed later in this section.

In this discussion, examples are also included of systems (such as P2P or protocol-based systems) which are not strictly Grid-based, but however employ methods that are relevant and applicable in a Grid environment. In order to provide a complete map of resource discovery, this review needs to include methods that are, even though not originating from a Grid perspective but are, potentially applicable to Grids and to provide a comprehensive analysis of the implementation effects of all such methods and to make comparisons on the basis of their analogy.

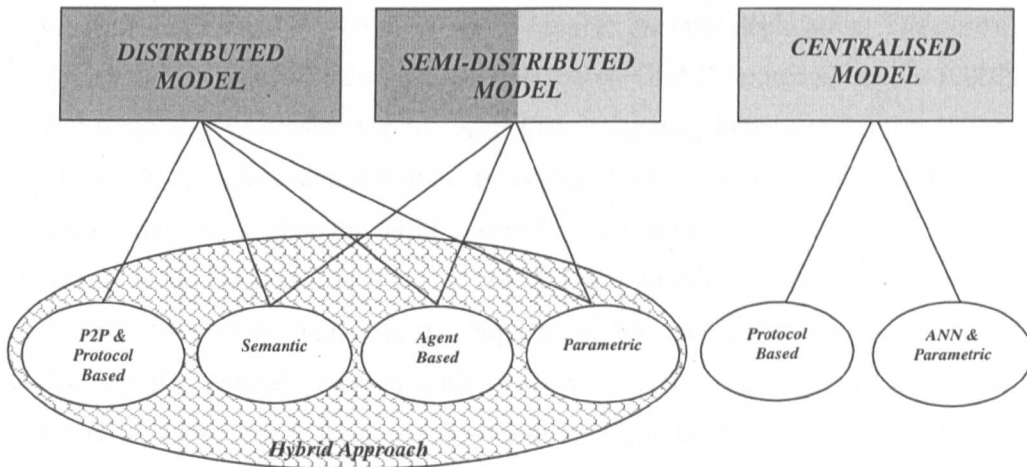


Figure 7: Taxonomy of Resource Discovery Methods

### 3.5.1 The Centralised Resource Discovery Model

In centralised resource discovery, there is a single hosting site which acts as a central repository for hosting complete information about the entire Grid resources. This hosting site could be a single computer or a cluster of computers collectively operating as a central server. The resource information and various sharing policies reside at this centralised point. Whenever a new resource is added or an existing resource is deleted/removed or modified, information on the central server's resource directory is updated. The client nodes or units, which send their requests for some particular resource, can access this information. Since there is centralised control, the entire network is dependent on a central site whose failure will inevitably cause the entire network to crash.

The centralised resource discovery model has been used to develop Grid-enabled resource discovery systems employing various approaches such as the Artificial Neural Network (ANN) & Parametric approach and the Protocol-Based approach.

#### *a. The ANN & Parametric Approach*

The ANN & Parametric approach was used in (Chen et al., 2004) to develop an Application-Oriented Grid Resource Discovery Service, which enables the users

to dynamically discover Grid resources suitable for their application. The core of this service is an artificial-neural-network-based Grid Resource Classifier (GRC) that periodically accesses the so-called Metacomputing Directory Service (MDS-1), which was a key component of the Globus Toolkit (Globus project, 2007) and dynamically classifies the Grid resources into application-oriented categories according to the real-time state of the Grid computing environment. Users can invoke this service and pass the application type as a parameter in order to discover the currently most suitable Grid resource. The topology of the ANN of the GRC uses several sigmoid units (neurons). The Globus Resource Allocation Manager (GRAM) (Globus Alliance, 2007) also can interact with this service to improve its practicality and efficiency. However, issues still not addressed are emulation, training of the ANN algorithm, time complexity of the training process and space complexity of the instance space of the ANN-based GRC. Moreover, MDS-1 is an older version of this service, whereas later versions such as MDS-2, MDS-3 and MDS-4 are not based on centralised model. These are described further in the section.

#### ***b. The Protocol-Based Approach***

Although not strictly a Grid-based system, Napster (Saroiu et al., 2002) follows the centralised resource discovery approach by using a large cluster of dedicated central servers, which maintain an index of the files that are concurrently being shared by active peers. Each peer maintains a connection to one of the central servers, through which file location queries are sent. The servers then cooperate to process the query and return a list of matching files and locations. On receiving the results, the peer may choose to initiate a file exchange directly from another peer. In addition to maintaining an index of shared files, the centralised servers also monitor the state of each peer in the system, keeping track of metadata such as the peers' reported connection bandwidth and the duration that the peer has remained connected to the system. This metadata is returned with the results of a query, so that the initiating peer has some information to distinguish and access possible download sites.

The Condor matchmaker (Raman et al., 1998) is based on the centralised approach, but does not use global names for resource discovery. Request queries for resources are sent by the Condor matchmaker to a central repository, the Condor collector, which is responsible for performing matching of resources. However, it does not address the issue of Quality of Service (QoS) for the discovered resources.

Globus's Metacomputing Directory Service (MDS-1) (Foster and Kesselman, 1997), (Globus Alliance, 2007) was also based on the centralised resource discovery method and uses the Protocol Based approach to discover resources on Grids. It is a single, unified access mechanism for a wide range of information sources. It uses the Lightweight Directory Access Protocol (LDAP) (Thompson, 2000), which is an open protocol standard supporting methods to manipulate data stored in network directories and comprises of four models; namely, *information*, *naming*, *functional* and *security*. It allows querying and manipulating information that exists in the Directory Information Tree (DIT). Building on the data representation and application programming interface defined by the LDAP, MDS defines a framework in which the information of interest can be represented in distributed computing applications and comprises of two components: the Grid Index Information Service (GIIS) and the Grid Resource Information Service (GRIS). Information is structured as a set of entries, where each entry comprises zero or more attribute-value pairs. The type of an entry, called its "object class", specifies mandatory and optional attributes. However, in the centralised approach the entire registry is hosted onto a single site, therefore the issue of scalability is most pertinent and is one of the major drawbacks of this implementation since there is a single point of (total) failure. Therefore, the MDS-1 was moved to a decentralised service and is now called Monitoring and Discovery Service (MDS-2) (Johnson and Brooke, 2002), (Globus project, 2007). This system consists of three distinct components namely, Representation and data access, Data model and Implementation (Fitzgerald et al., 1997).

### 3.5.2 The Distributed Resource Discovery Model

In distributed resource discovery, the resource information is dispersed across different sites. These sites could be a single computer or peer, each operating as a server or a cluster of peers collectively operating as server. Each peer is hosting the directory of its local resources and an index or link to the resource registry of other peers. Whenever demand to discover a specific resource arises, the search query is sent to the immediate peer, and if a match is not found then it is forwarded to the second nearest peer - if still not found then again to the next peer and so on. So each peer can send a resource request query and each is “surfing” and checked for resource availability. Since there is no centralised control, failure of any peer(s) does not affect the network in a catastrophic way. The distributed resource discovery model has been employed to develop Grid-enabled resource discovery systems by using various approaches such as P2P & Protocol Based, Parametric, Agent Based, Semantic and Hybrid approach.

#### *a. The P2P & Protocol-Based Approach*

Although P2P networks are not strictly Grid systems, several P2P approaches are relevant and applicable in a Grid environment. This is a type of distributed resource discovery method where each site is an independent peer. Peer-to-peer networks allow individual computers to communicate directly with each other and to share information and resources without using specialized ‘servers’ (Ripeanu, 2001). In a way, the Grid architecture is quite similar to P2P architecture – in fact, Grids are multi-peer to multi-peer.

In the P2P and Protocol based approach the query for resource discovery is broadcasted to all peers at the same time or to the immediate peer and then to others in a chain manner. To perform some particular task, specifically designed protocols are sent to various peers for efficient resource discovery. Each protocol has a dedicated functionality and some have been customized to perform enhanced/bespoke functionalities. Moreover, in protocol based resource discovery



systems, the Grid network grows by sending and receiving protocols, since gradually each node comes to know about all the other nodes on the Grid network.

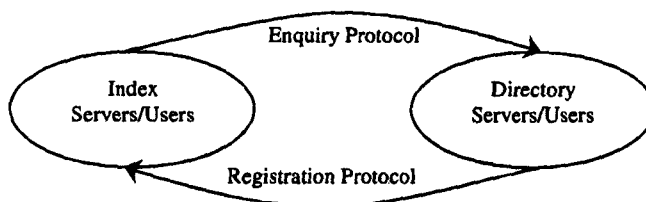
Various resource discovery algorithms in distributed networks are discussed in (Harchol-Balter et al., 1999), which proposes a randomized resource discovery algorithm called Name-Dropper, whereby all machines learn about each other within  $O(\log^2 n)$ -complexity rounds with high probability. Name-Dropper requires relatively few rounds and low network communication and achieves near-optimal performance both with respect to time and network communication complexity. However, it is assumed that the network is static, with no machines being added or removed when the algorithm is running, which is not always a valid assumption. Moreover, in some rounds it is possible that many machines in a network might choose at the very same time to contact the same one particular host which could only maintain a small number of simultaneous connections and hence would deny access to all other machines that are trying to contact it, severely restricting the access to Grid resources.

Other resource discovery algorithms have been proposed in (Law and Siu, 2000), (Zang et al., 2004) and (Iyengar et al., 2004). The work presented in (Kutten et al., 2001) is an extension of (Shiloach and Vishkin, 1982) and is also related to (Harchol-Balter et al., 1999), contributing with improved efficiency and message and time complexities.

Although not strictly a Grid-based protocol, Gnutella (Ripeanu, 2001) is an open, decentralised, P2P search protocol that is mainly used to find and share files. It employs the P2P & Protocol based approach, together with an aggressive flooding algorithm, to locate resources (files in this case). Computers running Gnutella protocol-compatible software form an application-level network and periodic ping/pong messages are used to propagate node information. In a Gnutella network, each node maintains open TCP connections with at least one other node, thus creating a virtual network of servants at the application level. Query and group maintenance messages are propagated using a flooding technique, while

query reply messages are back-propagated. However, error tolerance is a major issue in this technique, since it comes at a high price.

Another protocol-based approach using a request forwarding algorithm in a Fully Decentralised Grid Environment is presented in (Iamnitchi and Foster, 2002), where four types of request forwarding algorithms, namely random, experience-based+random, best-neighbour, and experience based+best neighbour are tested, keeping the resource frequency (number of resources) constant on an emulated Grid by using a membership information protocol to define the connection graph that changes over time. The technique comprises of an index server or users which send an enquiry protocol to the directory servers and the directory server or sources that send a registration protocol to the index server (Figure 8).



**Figure 8: Protocol Messaging Between Servers/Users**

Comparisons are made by sending independently generated sets of 200 requests to a set of randomly chosen 10 nodes repeatedly with the same set of requests and nodes. Instead of filenames, resource attributes are passed as parameters in the query, hence the requests specify sets of desired attributes and values. The proposed mechanism could be used to associate entities into directories and organize these directories into flat, dynamic networks. Results showed that the experience-based+random algorithm performs best in all request distributions, but requires more storage space and hence is more expensive than the random algorithm which is the least expensive, albeit also the least efficient. One of the limitations of this framework is the uneven spread of information; only nodes contacted by users learn, while other nodes remain uninformed and inexperienced. Moreover, instead of having real user logs, two request distributions - namely random and geometric - are chosen to match the requests with existing resources.

The number of distinct requests in the random distribution is approximately twice as large as that in an equally-sized geometric distribution. The technique used is quite unrealistic in the sense that all resources in a Grid are considered to be equally common. It is assumed that the storage space for logs is infinite and there are no failures, which is not true in a real world Grid.

An extension of (Iamnitchi and Foster, 2002) is proposed in (Iamnitchi and Foster, 2004), which is a general resource discovery solution, where the four types of request forwarding algorithms are used with four newly defined dimensions of the solution space namely, *membership protocol* that refers to how new nodes join the Grid and learn about each other, *overlay construction* that selects the set of active collaborators from the local membership list, *pre-processing* that refers to the offline preparations for better search performance, independently of requests, and *request processing* that searches or maps the local and remote resources according to the request. This study also describes four environmental parameters that influence the performance and design of a resource discovery mechanism namely, *resource information distribution and density* which refers to the fairness of sharing, *resource information dynamism* which refers to the dynamic and static resource attributes, *requests distribution* which refers to the pattern of user's requests for resources and *peer participation* which refers to nodes joining and leaving the network. This study claims that the proposed four components can define any decentralised resource discovery design. As a result, a simple resource discovery mechanism based on request propagation is evaluated, and the results are similar to those of (Iamnitchi and Foster, 2002).

A P2P based approach is presented in (Andrzejak and Zhichen, 2002), in which the CAN-based DHT system (Content Address Network-based Distributed Hash Table) has been extended into an indexing infrastructure which allows querying of ranges and supports efficient handling of dynamic data by using the so-called Space Filling Curve, especially the Hilbert Curve, as hash functions. In the CAN based P2P network (Ratnasamy et al., 2001), a subset of the servers participating in the Grid will act as nodes and store the pairs (attribute-value, resource-ID).

Each of them is responsible for a certain subinterval of the attribute values. Such a server is known as Interval Keeper (IK) and the corresponding subinterval its interval. Each server in the Grid reports its current attribute value to an IK with the appropriate interval. However, this strategy might become less efficient if the number of IKs grows and equivalently the sizes of their intervals decrease.

NEVRLATE (Chander et al., 2002) (a scalable resource discovery method) for an efficient organization of directories or directory mirrors, providing a scalable distributed resource discovery service where nodes are divided into several groups. It supports expressive lookup mechanisms and the directory servers are organized in an approximate two-dimensional Grid, or a set of sets of servers, where registration occurs in the 'horizontal' dimension, and lookup occurs in the other 'vertical' dimension. It focuses on distributed directories for distributed resources; therefore resource information is disseminated to all groups. However, it is assumed that resources are fairly stable, which is not always a valid assumption. Moreover, the overhead cost of lookup and publication is too high.

Another resource discovery model is Routing Transferring (Li et al., 2002), which defines three basic elements; namely, resource requester, resource router and resource provider. The resource information sent by the provider to a router is maintained in "routing tables". When a resource request sent by the requester is received by the router, then it checks the routing tables to choose a route for it and transfer it to another router or provider. The complexity of the proposed SD-RT (Shortest Distance Routing-Transferring) algorithm is analysed. The analysis shows that the resource discovery time depends on topology (the longest path in the graph) and the distribution of resources. When topology and distribution are definite or defined, the SD-RT algorithm can find a resource in the shortest time. The performance is determined by resource frequency and resource location. Moreover, high frequency and location of resources can reduce the resource discovery time significantly.

A distributed-index mechanism, called Routing Indices (RI), is presented in (Crespo and Garcia-Molina, 2002). RI maintains indices at each node and allows nodes to forward queries to neighbours that are more likely to have answers. If a node cannot answer a query, it forwards the query to a subset of its neighbours, based on its local RI, rather than by selecting neighbours at random or by flooding the network by forwarding the query to all neighbours. Three RI schemes are presented; namely, *the compound*, *the hop-count*, and *the exponential routing indices*. Results show that RIs can improve performance by one or two orders of magnitude vs. a flooding-based system, and by up to 100% vs. a random forwarding system. However, the exponential RI has the assumption that the network topology is a regular tree and that documents are uniformly distributed (the regular-tree cost model) which may not be realistic in some configurations.

Freenet (Clarke et al., 2001) is another file sharing system like Gnutella, which is not a Grid-based system, but shares files as the main resource and uses the request forwarding algorithm and cryptographic protocol. No broadcast search or centralised location index is employed. It is a location-independent distributed file system that provides an effective means of anonymous information storage retrieval and makes it impossible to discover the true origin or destination of the file passing through the network. Files in Freenet are identified by binary file keys obtained by applying a hash function (currently used function is 160-bit SHA-1). Three different types of file keys are also used; namely, *keyword-signed key (KSK)*, *signed-subspace key (SSK)*, and *content-hash key (CHK)*. Upon receiving request, a node first checks its own store for the data and returns it if found, if not found, it looks up the nearest key in its routing table and forwards the request to the corresponding node. If this request is successful then the node will pass data back to the immediate requester, cache the requested file in its local data-store and create an entry in its routing table associating the actual data source with the requested key. A subsequent request for the same key will be immediately met by the local cache, thus the routing tables are always dynamic. Hence file sharing is achieved by combining informed request and automatic file sharing. However, it

does not intend to guarantee permanent file storage, i.e. if the request for any specific file is not received for a long time then the entry is removed.

Chord (Stoica et al., 2003) is a scalable distributed P2P lookup protocol, which is also not Grid-based but addresses the problem of efficiently locating the node that stores a particular data item in a dynamic P2P system with frequent node arrivals and departures. Given a key, it provides distributed computation of hash function mapping keys to nodes responsible for them. In an  $N$ -node system, each node maintains information only about  $O(\log n)$  other nodes and a lookup requires  $O(\log N)$  messages to other nodes. Updates to routing information when a node joins or leaves requires  $O(\log^2 N)$  messages. Each node in the network hosts part of the index, and queries are hashed to create a key that is mapped to the node with the matching identifier. In Chord, nodes are organized in a ring. Each node maintains a small finger table that is used to forward queries around the ring until the correct node is located. However, the cost of a Chord lookup grows as the log of the number of nodes and its performance is degraded when a node's information is only partially correct.

A Virtual and Dynamic Hierarchical Architecture is proposed in (Lican et al., 2003) which employs an overlay network topology for discovering Grid services with high performance. Two service discovery algorithms - namely, Full Search Query and Discovery Protocol (FSQDP) - are also proposed that discover the node matching the request message from all nodes. There is no need for nodes to know all global names of groups or node identification, etc., because the groups are organized as a virtual group tree and the group and node properties can be obtained for example by the Query and Discovery Protocol. However, in a dynamic P2P environment it is hard to build and maintain global hierarchical topologies (Zhu et al., 2004).

Another P2P based approach using resource taxonomy to improve the efficiency of Grid resource discovery is presented in (Zhu et al., 2004), where application-level scheduling of resources is performed and resources are discovered according

to their attributes rather than IDs. It is a Resource discovery system that supports efficient attribute-based resource naming and query, for this purpose the concept of Resource Information Community (RIC) is introduced where resources are organized in communities. Just like nodes being grouped according to common interests in file sharing P2P networks, in RIC Grid information nodes with the same type of resources are grouped to resource information communities, and efficient navigation is supported via a DHT (Distributed Hash Table) P2P based bootstrap network. Various request forwarding strategies can be used to propagate requests inside the community such as flooding, random walk, etc. RIC-based resource discovery does not specify detailed node organization and request processing protocols inside the community, which provides flexibility for each community to adopt most appropriate protocols of its own. Routing Transferring (RT) is adopted inside each RIC and flooding is used to forward requests. However, it is assumed that each resource can be classified into a single type only, whereas it is possible for a resource to fall under multiple types at the same time, which is not always a valid assumption. Moreover, there is a huge topology construction and maintenance (C&M) overhead of each information node.

The so-called Non-uniform Information Dissemination protocols are proposed (Iyengar et al., 2004), to efficiently propagate resource information to nearby repositories without requiring flooding or centralised approaches. Two new protocols introduced for dynamic information dissemination are the *Change Sensitive Protocol (CSP)* that filters out information and prevent it from being disseminated if it changes too quickly or too slowly and the *Prioritized Dissemination Protocol (PDP)* that allows resources to be separated into priority classes, with different forwarding policies implemented for each. The non-uniform dissemination of resource information is used to reduce the overhead of uniform information replication, while maintaining accurate information at locations where it is most likely to be needed. The overhead in starting the job and transferring data and results increases as the distance from resources increase. Resource information is disseminated with a frequency and resolution inversely proportional to the distance from that resource; i.e. the nearer the resource, the

more information nearby repositories have about it and vice versa. The dissemination protocols work by propagating the resource state information more aggressively and in more detail to nearer information repositories than they do to the farther ones. Thus, repositories have more accurate and fresher information about nearby resources, but less accurate and less fresh information about distant resources. Results indicate a significant reduction in the overhead compared to uniform dissemination to all repositories. However, the criteria for propagating information non-uniformly could still be improved by an evaluation that would include a wider range of topologies and better information dissemination overlay backbones. Promising approaches include, for example, allowing query patterns or a resource's similarity to other resources or resource utilization to determine forwarding probabilities. Alternatively, the query success rate can be used. The development of such criteria can benefit from resource and query traces from a realistic large-scale Grid environment, but unfortunately, such information is not currently available.

Efficient Resource Discovery in Grids and P2P Networks (Antonopoulos and Salter, 2004) is a distributed approach for resource discovery that utilizes a small number of messages for query processing and building the network by distributing the inverted index over many network nodes replicating information and using a preference list. Each node represents a group of one or more machines connected to the network and hosting resources. The resources are registered against one or more keywords describing them in a local resource table on their local node. A node may become a supernode responsible for one or more keywords and maintain a node keyword table. Each node hosts a local node lookup table that contains a list of supernode-keyword pairs, i.e. determines which supernode is responsible for which keyword. A pointer in each supernode is introduced which points to the newest supernode added or created. Supernodes are connected together on a timeline and certain supernodes along this timeline are designated as checkpoints. This approach eliminates query broadcasting by implementing distributed inverted index structures such as checkpoints, supernodes and timelines. However, failure of certain nodes could break the timeline, so extra



routing information must be added to give each supernode an increased knowledge of its environment.

Other P2P distributed (non-Grid) systems in which resources are treated as files and are identified through their names such as CAN (Content Address Network) (Ratnasamy et al., 2001), Pastry (Rowstron and Druschel, 2001) and Tapestry (Zhao et al., 2001), use intelligent positioning of data into search-optimized, reliable and flexible structures such as distributed hash tables (DHTs) for efficient and scalable name-based retrieval. They build search-efficient indexing structures that provide good scalability and search performance. However, it is achieved at an increased cost for file and node insertion and removal. An implicit assumption in these systems is node homogeneity – i.e. all nodes are expected to have the same capabilities (Iamnitchi and Foster, 2004). Moreover, DHT-based schemes cannot support efficient attribute-based resource discovery.

#### ***b. The Parametric Approach***

In this approach parameters are sent to the distributed nodes and, depending upon those parameters, the query is dealt with.

The Parametric approach has been employed in (Maheswaran and Krauter, 2000) to discover resources by associating higher value with nearby information and reducing the data dissemination overhead. The notion of “Grid potential” is introduced, which weights a Grid resource’s capability with its distance from the application “launch point”. The tradeoffs are studied between three different protocols namely, the *universal protocol* which attempts to disseminate information uniformly, the *neighbourhood protocol* which limits the scope of dissemination to nearby nodes and the *distinctive awareness protocol* which is intended for unique Grid resources. The idea of having different protocols for different types of resources is similar to the rationale of the Prioritized Dissemination Protocol (PDP) (Iyengar et al., 2004). Simple tests are used to measure message complexity (overhead) and dissemination efficiency (error), but

despite calling this method a “parametric approach”, only a single point in the space is explored (Iyengar et al., 2004). However, instead of using benchmarks for the Grid potential, application-based measurement strategies can be used and theoretical performance models for data dissemination algorithms that belong to the distinctive awareness category can be constructed.

### *c. The Agent-based Approach*

Autonomous and mobile software agents are widely regarded as necessary components of large-scale distributed systems. Agents can facilitate/grant access to existing services to thin clients, support nomadic computing, perform functions related to resource management, support negotiations among several parties involved in a transaction, reconfigure servers, and so on (Jun et al., 2000). Resources host services which are considered as agents.

Various resource discovery algorithms are compared in (Jun et al., 2000), which introduces an agent-based model for resource discovery which uses an algorithm and a framework for the dynamic assembly of agents that are capable of providing detailed information about distributed network resources. Agents running at individual nodes learn about the existence of each other by using a mechanism called Distributed Awareness. Each agent maintains information about the other agents it has communicated with over a period of time and exchanges this information among them periodically. On identifying the target system, the agent creates a description of a monitoring agent capable of providing the information about remote resources, and sends this description to the remote site. There an agent factory assembles dynamically the monitoring agent. The remote agent creation and surgery techniques are generic and allow altering drastically the behaviour of an agent. However, modelling and analysis of the distributed awareness algorithm is rather difficult.

Moreover, (Aversa et al., 2004) proposes the so-called Terminal-aware Grid Resource and Service Discovery and Access Based on Mobile Agents Technology

and deals with the utilization of Web services technology to discover and optimally access Mobile Grid resources and services, within a Mobile Agent based Grid Architecture (MAGDA) (Aversa et al., 2003). Within the MAGDA framework, resources and services are characterized by mobility features. A resource is defined as a node able to host a mobile agent. A service is an application server or a mobile agent. The user is able to discover available services or to start own services downloading the agent code or asking for the agent's creations. The agents are able to discover new hosting nodes in order to explore the network or to move to less busy machines, to look for required resources or application. The Web Services paradigm and SIP and UDDI (Microsoft, 2000) technologies are utilized to implement a resource discovery service that allow users and mobile agents to look for and access distributed resources and applications, through heterogeneous terminals, by dynamically configuring the interaction session and service functionalities based on the characteristics of the terminal and the QoS of the interconnection.

#### *d. The Semantic-based Approach*

This approach implements semantics and ontologies to define resources. Each resource must operate according to its machine-understandable semantics. For the efficient and effective correspondence among the various devices on a Grid, their semantic descriptions must possess a frequency matching that aims at eliminating all the platform compatibility issues. On a Grid, resources are both time and space shared. When a new resource is added onto a Grid, its semantics must be specified. Many applications are being developed for Grid resources discovery using this approach, such as DAML+OIL (Darpa's Agent Markup Language + Ontology Inference Layer) (DARPA Agent Markup Language, 2007), which is a recently developed and used ontology representation language. The semantic approach has certain advantages over the other approaches, since semantic matching is flexible and effective.

Grid-SD (Grid-Service Discovery) (Ludwig and Santen, 2002) proposes a service discovery framework for Grid environments which relies on an ontology description that allows semantic matching and is based on the LARKS matchmaker (Sycara et al., 1999). The matching mechanism comprises of three filter stages; namely, context, syntactic and semantic matching, whereas the service ontology database provides the knowledge base. It relies on DAML-S (DARPA Agent Markup Language services) (Ankolekar, 2001) and its ontologies for matchmaking. The advertisements must match the requests; both refer to DAML (The DARPA Agent Markup Language) concepts and the associated semantics. The service matchmaker mediates between service requesters and service providers for mutually beneficial cooperation. Each provider must first register with a registry, also known as the matchmaker. The service provider advertises their capabilities by sending some appropriate messages (advertisements) describing the kind of service they offer. Upon receiving a request, the matchmaker matches it with its actual set of advertisements. On successful match, the matchmaker returns a ranked set of appropriate service providers and the relevant advertisements to the requester.

A design and a prototype is presented in (Tangmunarunkit et al., 2003) for a matchmaker using existing semantic web technologies and exploiting ontologies and rules (based on Horn logic and F-Logic) for resource matching where both the resource and request descriptions are considered as asymmetric. Resource descriptions, request descriptions, and usage policies are all independently modelled and syntactically and semantically described using the Resource Description Framework (RDF), which provides data model specification and XML-based serialization syntax. The ontology-based matchmaker consists of three components; namely, *ontologies* which capture the domain model and vocabulary for expressing resource advertisements and job requests, *domain background knowledge* which captures additional knowledge about the domain and *matchmaking rules* which determine whether or not a resource matches a job description. Domain background knowledge captured in terms of rules is added for conducting further deduction. Finally, matchmaking procedures written in

terms of inference rules are used to reason about the characteristics of a request, available resources and usage policies to appropriately find a resource that satisfies the request requirements. Additional rules can also be added to automatically infer resource requirements from the characteristics of domain-specific applications, without explicit statements from the user. However, in the case of recursive rules, the evaluation may be time consuming.

One of the mainstream Grid technologies, OGSA-DAI (Open Grid Service Architecture Data Integration and Access) (Antonioletti et al., 2005), has been used quite intensively for the Semantic-based approach. A mediator-wrapper architecture and ontology based semantic information has been proposed (Tan et al., 2007) to wrap the heterogeneous data source. It employs a mediator structure to supply accessing interface for the data sources, and it builds virtual data source (VDS) to support standard OGSA-DAI query interface.

#### *e. Hybrid Approach*

Some network resource discovery systems have been developed by combining two or more distributed model approaches described above to optimize the efficiency and output of the system at hand.

A Hybrid approach is suggested in (Mastroianni et al., 2004), which combines both P2P and Semantic approaches in a sophisticated manner. It adopts a P2P approach for managing global queries on multiple Index Services. Metadata models are studied using an XML-based approach for heterogeneous resource representation and management in Grid-based data mining applications, especially in Knowledge Grid which is an extension of the work done in (Mastroianni et al., 2003). By using services, tools, and repositories provided by the two layers of the Knowledge Grid - namely, the *Core K-Grid layer* and the *High level K-Grid layer* - a user can search and identify data sources, data mining tools, and computational resources. Then all these components can be combined to build a distributed/parallel data mining application that can be executed on a Grid.

eXtensible Markup Language (XML) is used to represent metadata in XML documents according to a set of XML schemas defined for different classes of resources. This metadata is managed and accessed by a set of services defined on the two layers of Knowledge Grid. The information managed by the Knowledge Discovery Service (KDS), one of the services on the Core K-Grid layer, is stored into three repositories: the Knowledge Metadata Repository (KMR), the Knowledge Base Repository (KBR) and the Knowledge Execution Plan Repository (KEPR). *Predictive Model Markup Language (PMML)* is a standard framework that has been defined to describe data mining results.

A Hybrid approach in (Heine et al., 2004) uses an Ontology-Driven P2P Grid Resource Discovery system to address the issue of semantic resource discovery in Grids by using an ontology-based peer-to-peer search network to distribute and query the resource catalogue. P2P networking is used to distribute both the assertional and the conceptual knowledge. Each peer can provide resource descriptions and background knowledge, as well as query the network for existing resources. A central ontology for resource description and matching is not required. This means that the inherent incomplete ontology of any peer will be complemented by the knowledge of other peers distributed over the network. This allows the network to deduce answers to queries and find matching resources, tasks that could not have been possible by querying individual peers, as the network supplies the missing parts of the ontology.

Ontologies based on description logics are used to describe the resources. Information is distributed over a peer-to-peer network based on distributed hash tables. Thus it enables detection of resource matches, even if a provider within the Grid does not know all the terms used in a resource query. This has been achieved by combining the knowledge of all peers within a distributed classification DAG (Directed Acyclic Graph), so that queries can be resolved against this DAG. However, it is assumed that peers do not leave the network accidentally or without informing, which is always not a true assumption, since in a real world peers might break down and have to leave the network without notice. Therefore, some

issues to be addressed further are completeness, expressiveness of queries, fault tolerance, garbage collection, ranking of results and routing optimization.

Another stream of research is orientated towards achieving Grid resource discovery through “Services” (Grid and Web services). The Open Grid Service Architecture (OGSA) (Foster et al., 2004) defines the Grid service concept, based on principles and technologies from both the Grid computing and Web services communities (Talia, 2002). Moreover, OGSA not only defines the semantics for a Grid service, but also defines standard mechanisms for creating, naming, and discovering transient Grid service instances. It also provides location transparency and multiple protocol bindings for service instances and supports integration with underlying native platform facilities (Foster et al., 2002). Also, Grid services are no longer considered separate from the Web services. In fact, according to the Open Grid Service Infrastructure (OGSI) version 1.0 specification (Tuecke et al., 2003), a Grid service is considered to be a Web service that conforms to a set of conventions (interfaces and behaviours) which define how a client interacts with a Grid service for such purposes as service lifetime management, inspection, and notification of service state changes (Foster et al., 2005). These conventions, and other OGSI mechanisms associated with Grid service creation and discovery, provide for the controlled, fault-resilient, and secure management of the distributed and often long-lived state that is commonly required in advanced distributed applications. The MDS3 or Globus Toolkit 3.2 (GT3.2) is a software toolkit based on OGSI that can be used to build Grid-based applications.

Recently there has been a drift from OGSI to the Web-Service Resource Framework (WSRF) (Czajkowski et al., 2004b) due to potential performance advantage reasons. WSRF is concerned primarily with the creation, addressing, inspection, and lifetime management of state-enabled resources. It codifies the relationship between Web services and state-enabled resources in terms of the implied resource pattern, which is a set of conventions on Web services technologies. A state-enabled resource that participates in the implied resource pattern is termed a WS-resource. The framework gives the WS-resource definition

and describes its association with the description of a Web service interface. It also describes how to make the properties of a WS-resource accessible through a Web service interface and how to manage a WS-resource's lifetime. The MDS4 or Globus Toolkit 4 (GT4) is a full implementation of WSRF. Based on industry feedback, the revised and updated WSRF specifications are submitted to two new OASIS technical committees, the WS-Resource Framework (WSRF) TC and the WS-Notification (WSN) TC (Baker et al., 2005).

A high level OGSI-compliant hierarchical Information Service built on Globus Toolkit MDS-3 is presented in (Zang et al., 2004). Following the OGSA standard, it integrates with local information suppliers that are implemented as OGSI-compliant Grid services, such as local resource management systems, job Grid service, job queuing Grid service, etc., and supports information collection, update and accessing on a Grid Virtual Organization that consists of multiple administrative domains and resources. The proposed information model includes job status, computational resources, local resource workload, service metadata, and queue status. This information is further classified into two categories: the static information and the dynamic information.

Semantic Grid and P2P data integration has been studied (Zhou and Wang, 2006), in order to exploit their strengths in a common framework. The impact of P2P and the semantic Grid technologies has been investigated to steer Enterprise Information Integration (EII) systems to a new decentralised, flexible, scalable system, which would be compatible with OGSA-DAI.

Currently, a lot of work is being devoted to Grid resource discovery using web services. Each of these endeavours constitutes a step towards the integration of Grid services into Web services, which might provide a viable solution to the problem of resource discovery in Grids. This integration is necessary since Web services cannot directly be used as Grid services due to intrinsic limitations of Web services such as statelessness, etc. (whereas a Grid service must have a state since it is prone to dynamic changes and has more complex functionality than an



ordinary Web service). To serve the purpose of resource discovery, Grid services could be glued to Web services using specialized object-oriented techniques such as encapsulation or inheritance, etc.

### 3.5.3 The Semi-Distributed Resource Discovery Model

The semi-distributed resource discovery model combines centralised and distributed models into a consistent broker system which maintains the resource directory and registers each resource on the Grid. The broker is responsible for matching or assigning the right resource to the request query for resource discovery. The semi-distributed resource discovery model could also be employed to develop Grid-enabled resource discovery systems by using various approaches, such as Parametric, Agent Based, Semantic and Hybrid, as explained earlier in section 5.2.

EZ-Grid system (Chapman et al., 2001) aims at enabling efficient use of Grids by both end users and administrators. It uses a sophisticated brokering system coupled with usage policy framework and a distributed information subsystem to achieve user specified time/cost constraints and analyses static as well as dynamic information about resources. It has two main components; namely, *client component* and *server component*. The resource brokerage system uses the information subsystem, policy framework and the scheduler-specific interfaces to make resource choices based on the job specifications. The brokering process could be budget-based or deadline-based or both. The broker would then be expected to arrive at resource choices that would provide the quickest execution time or the least cost execution. Although it analyses dynamic resource-specific and scheduler-specific information and supports budget/deadline based scheduling, the entire process of arriving at appropriate resource choices based on resource status information and history information is a huge and complicated problem.

The Nimrod/G method (Buyya et al., 2000a) is a Grid-enabled resource management and scheduling system built on Nimrod. It follows a modular and component-based architecture enabling extensibility, portability, ease of development, and interoperability of independently developed components. It uses the Globus toolkit services and can be easily extended to operate with any other emerging Grid middleware services. The concept of computational economy is introduced as part of the Nimrod/G scheduler. The key components of Nimrod/G are: Client or User Station, Parametric Engine, Scheduler, Dispatcher and Job-Wrapper. The scheduler selects resources that meet the time and cost limits. The brokerage system relies on Globus GIS to gather information about remote resources. However, the cost changes as other competing tasks are placed on the Grid and increases complexity. Moreover, this approach can employ only a simulated model for investigation purposes due to the unavailability of middleware services.

Semantic Matching of Grid Resource Descriptions is proposed in (Brooke et al., 2004), which uses the semantics approach for resource descriptions. Ontologies are used in the Grid Interoperability Project (GRIP), which enables brokering for resources described by several Grid middleware systems: GT2, GT3 and UNICORE. The proposed broker is able to interrogate on behalf of its clients two different resource schemas. The GLUE schema (Andreozzi, 2004) is used to provide a uniform description of resources on the Data Grids being developed in the US and Europe and to enable federation of relevant VOs for global analysis of data from particle physics experiments. The other schema is provided by the UNICORE framework (Erwin and Snelling, 2001), a software model that creates local Incarnation Data Base (IDB) entries, used to 'ground' or 'incarnate' Abstract Job Objects (AJO), which are composed on behalf of client applications and sent around the Grid as serialized Java objects. However, this work supports only very small subsets of UNICORE and GLUE that can be immediately mapped in this way.

A resource broker focusing on matching the available resources to the user's requests is presented in (Afgan, 2004). It provides a uniform interface to access any of the available and appropriate resources using the user's credentials. The process of creating a resource broker is discussed and an insight into how it connects and relates to the underlying software is provided. The resource broker runs on top of the Globus Toolkit. Therefore, it provides security and current information about the available resources and serves as a link to the diverse systems available on the Grid. The user contacts the resource broker and sends a request by filling a web form on a simple webpage and specifying the request in very general terms. Upon receiving the request, the resource broker looks for a match by communicating to the GIS (Grid Information System) (Globus project, 2007), which returns requested information in plain text format, later converted into XML. The XML output is then parsed, extracting only fields that match the fields specified by the user in the request. The response is processed until it is determined that a match can or cannot be found. Although there is flow of information from the user and back to the user throughout the Grid, one thing that the resource broker does not provide is any sort of job scheduling among the resources.

### **3.6 Critical Analysis**

An advantage of the centralised resource discovery method is that its very simple and centralised architecture is easy to design, implement and maintain. Moreover, data management is easy since the entire data is hosted at a single point. But at the same time this is also a major drawback - the entire architecture is dependent on a single central node, causing it to have a single point of failure and lack of fault-tolerance. In fact, there should be less centralised control (Iamnitchi and Foster, 2004), because in the case of centralised control there might not be an incentive for any participant (institution or individual) to support the significant administrative costs inherent in systems that aggregate a huge number of resources with unpredictable behaviour. Moreover, since a centralised architecture is challenged both in terms of scale and dynamic behaviour (flexibility /

adaptability), the adoption of a self-configuring, distributed architecture would provide a more preferable solution. The centralised resource discovery model is less reliable because, if a service is terminated (due to system attack or system failure), then access to the entire network is denied and inevitably such a situation is difficult to remedy or manage. Scalability is also a major issue, since the centralised model cannot handle extraordinarily large number of nodes and resources. So if the network grows fast, then the efficiency of this model drops - this is one of the reasons behind the efforts to move Globus MDS-1 service from centralised form to the decentralised MDS-2 (Czajkowski et al., 2001).

By and large the distributed resource discovery model is more powerful than the centralised model. Resource discovery systems developed based on this approach are usually more successful than the ones developed using the centralised approach. However, still the issue of scalability remains, which makes it difficult to manage large volumes of resources on a vast network like a Grid. Moreover, Grid environments, being highly heterogeneous in nature, may make the use of DHTs ineffective in Grid-enabled resource discovery systems (Iamnitchi, Foster 2004), since in DHTs all nodes have equal responsibilities assuming homogeneous capabilities and trust. Moreover, in typical resource discovery systems the properties of resources or requests are based on symmetric flat attributes, which might become unmanageable as the number of attributes grows (Tangmunarunkit et al., 2003).

The semi-distributed resource discovery model has in many respects a more privileged architecture than both the centralised and distributed models. It can serve as a better option for creating resource/request brokering systems and would be better able to facilitate the design of middleware packages. However, it has some limitations as well. One of the limitations of the semi-distributed resource discovery model is complexity, as the demanding technical-level integration required makes it difficult to manage and maintain the integrity and consistency of the entire architecture. Moreover, fixing (repairing) the network could be very

time consuming and there can be a risk that at some nodes the problem of load-balancing may arise.

**Table 1: A Comparison of Different Systems surveyed that Taxonomises Various Semantic Integration Mechanisms used by them**

<i>Technology used for Semantic Matching</i>	<i>Level of Semantic Matching</i>	<i>RD Model used</i>	<i>RD Approach used</i>	<i>Reference</i>
DAML-S	Service-Level	Distributed Model	Semantic Approach	Grid-SD (Grid-Service Discovery) Ludwig and Santen, 2002
RDF	Resource-Level	Distributed Model	Semantic Approach	Tangmunarunkit et al., 2003
OGSA-DAI	Resource-Level	Distributed Model	Semantic Approach	Tan et al., 2007
XML	Resource-Level	Distributed Model	Hybrid Approach	Mastroianni et al., 2004
DAG	Resource-Level	Distributed Model	Hybrid Approach	Heine et al., 2004
OGSA-DAI	Resource-Level & Service-Level	Distributed Model	Hybrid Approach	Zhou and Wang, 2006
GLUE	Resource-Level	Semi-Distributed Model	Semantic Approach	Brooke et al., 2004

Table 1 taxonomises the semantic integration mechanisms used in the different systems surveyed. This taxonomy is based on certain important defining characteristics of these mechanisms such as: the technology used for semantic matching, the level of semantic matching, the Resource Discovery model used and the Resource Discovery approach used.

It is clearly seen from Table 1 that the level of semantic matching in all the systems surveyed is either at resource-level or service-level and none of these

systems performs semantic matching at the data field-level or attribute-level, which is needed to fully address the semantic heterogeneity challenge.

The proposed approach in ASIDS architecture is novel as it performs semantic matching at the data field-level and without using any of the complex semantic mapping tools, which is the unique characteristic of ASIDS.

### **3.7 Conclusion**

This chapter has presented review of the past and ongoing efforts to resolve the issue of resource discovery in Grids, based on complete taxonomies of Grid resources and resource discovery methods. The various resource discovery methods, techniques and approaches are discussed along with their advantages and disadvantages, and eventually recommendations are made with respect to practical implementations and directions of future research in Grid resource discovery.

Various approaches have been proposed and used to resolve the problem of resource discovery in Grids, based on three basic resource discovery models: centralised, distributed and semi-distributed.

The centralised resource discovery model is the simplest and a centralised architecture is easy to design. Moreover, data management is easy since the entire data is hosted at a single point. However, it is not likely to be recommended as a resource discovery solution in general purpose Grids, since a centralised architecture is entirely different from Grid architecture, which is multi-peer to multi-peer. Hence it would be an inappropriate route to follow in the case of ever-growing networks, due to the major issue of poor scalability. Furthermore, poor security and reliability are serious disadvantages of the centralised model.

The distributed resource discovery model can address this problem to some extent, but is not a perfect solution, since even though being distributed in nature,

if the network grows it becomes difficult to maintain and track the huge reservoir of resources, reducing the efficiency of a resource discovery mechanism. The issues of scalability and architectural compatibility arise in both the centralised and distributed models.

The semi-distributed resource discovery model can provide the best option for creating resource/request brokering systems and designing related middleware packages, since overall it seems to be more reliable. However, this model also has some limitations, such as complexity, time, costs and difficulty of managing/maintaining, etc. In order to provide optimal service, such systems need to be easily configurable (manageable), flexible and generic (reusable). Moreover, a semi-distributed network architecture should be modelled in a sophisticated manner, so as to address the scalability issue sufficiently to ensure that its effectiveness and efficiency remain unaltered regardless of the number of nodes or peers or resources added or removed from the network. It must also possess load-balancing and fault-tolerance features.

There are many different methods that have been used to address the issue of resource discovery and several approaches have been taken, yet a complete solution is not available. However, since Web services can be used to discover resources on the Grids, one way of achieving successful resource discovery is by integrating the Grid services with Web services or gluing them together. It seems that using a Hybrid approach over a Semi-Distributed architectural model can help resolve the problem of resource discovery in Grids to some extent. A further step for resolving the heterogeneous data federation issue has been taken by the advancement and implementation of OGSA-DAI. Although the potential and promise is there, it is clear that further advances are needed in this field in order to provide a satisfactory, all-round solution framework.

## **Chapter 4**

### **Taxonomy of HealthGrids: Types of HealthGrids, Resources and their Discovery in the Healthcare Domain**

#### **4.1 Introduction**

Healthcare is currently going through a series of technological advancements and modifications. Health information has always been of great importance to society and has a strong impact on various social aspects. Due to its nature, health information has to be dealt with great care and confidentiality. At the same time, it has to be shared and exchanged across various organizations or individuals to provide improved healthcare service. Two of the most important disciplines in healthcare today are bioinformatics and medical informatics. As Computer Science and Biotechnology communities join forces to create new technologies for the advancement of medical science and improvement of medical service delivery (Stewart, 2004), this means that more people will be able to lead normal, healthy lives.

It is widely recognised today that the healthcare industry requires customized solutions with respect to information integration. The information sharing techniques currently available are not sufficient to meet the requirements of an integrated health care system. The state of electronic information integration in healthcare lags noticeably behind other business domains such as banking, insurance and electronic commerce (Bilykh et al., 2003). There is a need for health information systems to be fully integrated with each other and provide interoperability across various organizational domains for ubiquitous access and sharing. Moreover, due to rapid progress of biotechnology an increasing number of life science databases are becoming available that are being operated and



managed individually (Tohsato et al., 2005). Many existing solutions still do not offer the desired levels of utility/functionality or sophistication that a health information system demands.

The emerging technology of HealthGrids holds the promise to successfully integrate health information systems and various healthcare entities, including human and non-human, such as scientists, scientific tools, medical instruments, physicians, patients and all types of healthcare data, etc., onto a common (global) platform that would be shared and easily accessible. In such a scenario, each health information system is composed of various distinct components, which are integrated in a way that each component has its well-defined semantics and ontology and is well-aware of all other components.

This chapter is concerned with the problem of Resource Discovery in HealthGrids, which is an emerging challenge comprising many technical issues, such as performance, consistency of data/information, efficient retrieval of resources, compatibility of platforms, integrity of medical data, aggregation of storage resources and security of life-critical data, etc. Moreover, the quality and security of health-related data available on HealthGrids has to be significantly high, since this data may often be 'life sensitive'.

This chapter offers a systematic taxonomy of the HealthGrids and their resources. It first outlines the characteristic features and functionalities of HealthGrids, and reflects on the need for Grid technology in healthcare. A taxonomy of HealthGrids is proposed, based on their functionality, purpose, and application area. This chapter also proposes a taxonomy of HealthGrid resources and discusses the issue of resource discovery in HealthGrids. Considering the challenge of resource discovery, it discusses the problem of heterogeneity, issues of medical coding and terminology, and the role of semantic technologies; and it proposes potential solutions for the discovery of different types of HealthGrid resources. Finally, it reflects on discovering and integrating data resources and the future of HealthGrids and draws some conclusions.

Although this chapter explores the resource discovery challenges for all types of Grid resources such as storage, computation, etc., our study is focused primarily on the data and information type resources due to the complexity and large volumes of data available on the HealthGrids, which are a type of Grids.

## **4.2 Introduction to HealthGrids**

A HealthGrid is a Grid used in the context of healthcare. Based on the literature survey, we attempt to offer an updated and comprehensive definition of a HealthGrid:

*“HealthGrid is a Grid infrastructure dedicated to the management of healthcare resources that encompasses and integrates the various Grid components and healthcare components with consistent, compatible and meaningful coordination among them, to facilitate provision of the healthcare services.”*

HealthGrids are expected to possess enhanced, customized capabilities and features, such as:

- i. Remote access services
- ii. Common distributed databases for healthcare
- iii. Information sharing
- iv. Integration of heterogeneous information from disparate sources
- v. Common/standardized storage mechanisms
- vi. Efficient computation & data retrieval
- vii. Large-scale data processing
- viii. Shared access to computing resources
- ix. Social healthcare services
- x. Provision of secure access to:
  - a. Patient's medical history
  - b. Medical Images (e.g. mammograms)

- c. Standard formats of files & information for comparison
- d. Library of examples for training & diagnosis
- e. Health support services
- f. Drug details & clinical trials
- g. Health information systems

Many Grid projects related to healthcare are currently running across Europe, such as CrossGrid (CrossGrid.org, 2004) that targeted parallel (MPI) computing and interactivity, DataTAGrid (DataTag, 2007) which focuses on interoperability, and DataGrid (Breton et al., 2003) which is a prototype of the BioMedicalGrid and supports better medical record management and improved diagnosis. Other projects such as the MammoGrid (MammoGrid, 2007), GEMSS (Jones et al., 2004) and e-Diamond (Brady et al., 2003) for UK, the NDMA (National Digital Mammography Archive, 2007) for US and MEDIGrid (Boccia et al., 2005) for France are also in their completion phase.

An evolutionary cross-platform model is proposed in (Ruotsalainen, 2004), which is a Grid-like peer network that dynamically connects national security domains for the integration of purely internet-based health information systems. However, this Grid-like model lacks many features and functionalities, mentioned earlier in this section, that a pure HealthGrid possesses. HealthGrids are designed and used specifically in the contexts of clinical use and/or epidemiological studies, where data integrity and platform compatibility are necessary to provide consistent medical information to the various stakeholders of healthcare. These stakeholders include health specialists (doctors, physicians, and practitioners/ surgeons), medical lab technicians, pharmacists (drug developers, analyzers), health analysts, medical equipment providers/manufacturers, healthcare organizations and even patients or the general public. All of them need a globally interoperable channel to be able to carry out collaborative work on healthcare problems, and in one way or another, HealthGrids will have a beneficial impact on their everyday practice. Several BioGrid projects are running, such as the TeraGrid (TeraGrid Project,

2007) and the myGrid project (myGrid, 2007), which is one of the most well developed Life Sciences Grid projects in Europe.

Hence, work is still going on and efforts are being made towards a modernized facet of future healthcare by using HealthGrids as depicted in a prognosis for year 2013 (Silva and Ball, 2002).

### 4.3 HealthCare Needs Grid Technology

The case for the use of Grid technology in healthcare arises mainly from the need to improve, safeguard and effectively exploit the available *life-significant medical information*, the need to protect the privacy of *personal, life-sensitive health information*, and the need to provide *integrated healthcare services* and have in place effective, global *channels of collaboration*.

*Health-related information* is important for the well-being of society and has to be accurate and consistent. Medical information provided over the internet often suffers from ambiguity and contradiction that would increase the complexity and confusion of medical issues instead of solving them. Moreover, anyone can publish or post material of their choice over the internet without any peer review or checking, which makes open internet an unreliable source of healthcare information.

Information available on HealthGrids can initially be peer reviewed once before uploading, but even more importantly, it can be constantly and continuously checked and revised appropriately, thus making HealthGrids an accurate and reliable source of health information that can be accessed any time from any place.

“One of the major challenges faced by the biomedical research community is how to access, analyse, and visualize heterogeneous data in ways that lead to novel

insights into biological processes or that lead to the formulation of a hypothesis that can be tested experimentally” (Blake and Bult, 2006).

To exploit effectively the wealth of medical information, there is an urgent need to *integrate, manipulate, process, and analyse* huge heterogeneous datasets from disparate sources. More systematic use of Grid technology in healthcare will not only help meet the current needs for data processing, but will ensure that future demand for even more capacity to deal with far larger volumes of data can be met.

Moreover, whenever *confidential medical information* is shared among health organizations, security and privacy are critical issues (Bilykh et al., 2003), since HealthGrids contain ‘life sensitive data’. The information content in a healthcare system is related to various entities, such as hospitals and their staff, stakeholder organisations and their members, medical equipment/devices, medicines, diseases, information records, etc. Amongst all the entities, the *patient record* is the most prominent, since it encapsulates information on most other entities (some of which is personal, and should be kept private to the patient).

The patient record encapsulates instances of various other entities, all of which are pooled to make a complete EPR (Electronic Patient Record) for each patient. The EPR needs to be robust, so as to maintain the consistency of authentic health information.

On another level, HealthGrids can prove to be an *effective channel for international collaborations* where the world’s scientific minds can collectively work, such as to conduct a group-wise analysis, and might produce solutions that would effectively address complex medical problems (for instance, a disease or remedy).

There are many *other reasons* why the healthcare industry needs Grid technology:

- a. To provide more computational power
- b. To make network resources readily available
- c. To effect better use of system resources, and reduce waste by eliminating idle resources
- d. To create new business opportunities and exploit economies of scale
- e. To enable faster problem solving
- f. To support multiple operations by concurrent and ubiquitous access
- g. To provide the massive data storage spaces required in healthcare
- h. To make healthcare solutions/systems more efficient

All of the above and many more emerging issues demand to be addressed in a sophisticated manner by an advanced and reliable solution.

A study in (Estrella et al., 2007) discusses that Grid computing holds the promise of harnessing extensive computing resources located at geographically dispersed locations that can be used by a dynamically configured group of collaborating institutions. It defines a suitable platform on which distributed medical informatics applications could be based. Particularly, Grids can address issues relevant to medical domains such as data distribution, heterogeneity, data processing and analysis, security and confidentiality, standardization and compliance, etc.

HealthGrids are a good way to address these needs and provide reasonable solutions the challenges of modern healthcare. A study in (Piggott et al., 2004) explores the potential use of Grid technology in Healthcare, such as integration of heterogeneous data sets from multiple diverse sources systems. Thus, if successfully implemented, the HealthGrid will have a high impact towards lower costs and greater benefits for healthcare in the long run. In this respect, the HealthGrid could be the driver of the next generation of healthcare IT.

#### 4.4 Taxonomy of HealthGrids Types

There are various types of HealthGrids defined in the healthcare sector. Each has been devised for a dedicated purpose, so as to provide special services and to support the performance monitoring of specialized tasks in a particular healthcare sector. A taxonomy of HealthGrids types is proposed based on their functionality, purpose, and application area.

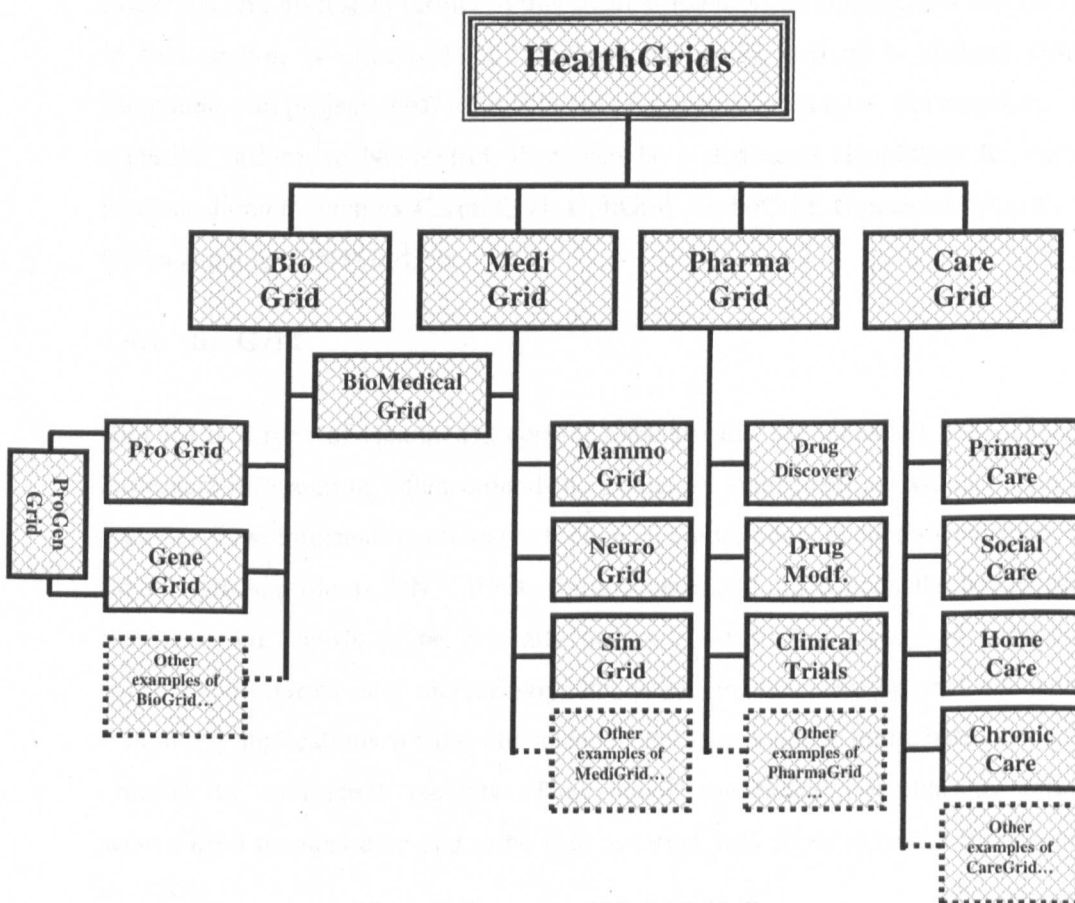


Figure 9: Taxonomy of HealthGrids Types

Figure 9 depicts the taxonomy of HealthGrids into four major types, namely BioGrid, MediGrid, PharmaGrid and CareGrid, where the BioGrid and the MediGrid merge into the BioMedicalGrid, which combines the features and functionalities of both Bio and Medi Grids. The BioGrid is sub-categorized into representative examples such as ProGrid for Proteomics and GeneGrid for

Genomics, both of which merge into the ProGenGrid. The MediGrid is also further sub-categorized into typical implementations, such as Medical Imaging (Visual) grids (e.g. MammoGrid) for the management and processing of medical images, scans or DICOM files, NeuroGrid for neurologists and SimGrid for medical simulations and modelling (another example of a Visual Grid).

The various types of HealthGrids, along with characteristic applications, are examined and discussed further in this section. Each of the HealthGrids described in this section is effectively a DataGrid and could also be a SemanticGrid (Semantic grid project, 2007) if it is based on semantic principles. For example, in a similar fashion to NeuroGrid, there can be a dedicated HealthGrid for each medical domain, such as CardioGrid, OptiGrid, OrthoGrid, GynaecologyGrid or Otolaryngology/ENTGrid, etc.

#### **4.4.1 BioGrid**

BioGrid is a type of HealthGrid designed specifically for accessing and sharing biological information, often around the globe, by authorized individuals and/or organizations. Information related to biological components at the molecular level such as genes, proteins, DNA, RNA, chromosomes and other molecular biological structures, etc. needs to be critically analysed for further biological research purposes. BioGrids are increasingly important in the development of new computing applications for the life sciences and in providing immediate medical benefits to individual patients. They have significant potential to offer personalised medical care and to be able to target only those at risk (Ellisman et al., 2004).

OGSA-DAI has been used in ChemBioGrid by bringing Data Management tools into collaborative environment. The mechanism has been studied, for supporting Digital Libraries in High-Performance Computing environment based on Grid technology. OGSA-DAI has been implemented to provides abilities to assemble heterogeneous data from distributed sources into integrated virtual collections



(Zhuchkov et al., 2006). The BioGrid is sub-categorized into the ProGrid and the GeneGrid, which are also the applicable examples of BioGrids and are described below.

#### *a. ProGrid*

ProGrid is a practical example of BioGrid that is specialized in the management of all types of information related to proteins, such as proteomic and proteo-type data, protein structures, protein identification, protein analysis, protein expression level, protein mutation, protein screening and classification, etc.

The human body is incredibly complex and consists of roughly 50 trillion cells, each consisting of an enormous number of components (of the order of  $10^{13}$ ), many of which are proteins. It normally takes months on a Peta-flop class computer (one capable of performing  $10^{15}$  calculations per second) to simulate the activity of a single protein, taking into account each atom in the protein. No such computer systems exist today, and designing one remains a formidable challenge (Stewart, 2004).

The ProGrid will be able to address this issue by making available enormous computation resources for highly complex computational operations. A recent study (Cannataro et al., 2005) presents MS-Analyzer, a tool for the management processing and analysis of proteomic Mass Spectrometry data. It is a specialized version of PROTEUS (Cannataro et al., 2004), which is a Grid-based Problem Solving Environment for bioinformatics applications that uses (a) domain ontologies to design complex in silico experiments by modelling basic software tools, data sources and workflow techniques and (b) data mining software tools to provide proteomics facilities. Its main requirements include interfacing with proteomics facilities, storing and managing proteomic Mass Spectrometry data, and interfacing with off-the-shelf data mining and visualization software tools.

An architecture combining the use of OGSA-DAI, Grid distributed querying (OGSA-DQP) and data integration software tools to support distributed data analysis has been proposed (Zamboulis et al., 2006), for the integration of several autonomous proteomics data resources

***b. GeneGrid***

GeneGrid is another practical example of BioGrid that is specialized in the management of all types of information related to genes and of relevance to genomic studies, such as information on genomes & genotype, genetic structures, genetic sequences, genetic mutations, genetic diseases, genetic epidemiology, gene therapy, gene naming, genetic analysis, gene screening, genetic variation and genetic classification, etc. For the purposes of genetic epidemiology GeneGrid can support the unified naming of phenotypes and standardised acquisition and recording of clinical parameters. In genetic epidemiology studies, a clinical annotation service is one of the central services in a Grid for clinical phenotype descriptions (Breton et al., 2005). The GeneGrid project (Jithesh et al., 2005) integrates numerous bioinformatics programs and databases available on different resources across various sites allowing scientists to easily access the diverse applications and data sources without having to visit many web servers. This reduces the overall time for executing the experiment. The Grid services developed in the GeneGrid project are based on the Open Grid Services Architecture (Foster et al. 2003) and provide scheduled access to resources, data, and applications, using XML-based messages.

The need to have a dedicated GeneGrid arises due to the ever-increasing volumes of genomic data and ever more demanding complex computations for genetic operations.

### **c. ProGenGrid**

The ProGrid and GeneGrid merge into the ProGenGrid which is dedicated to perform management of data related to the sequence and structure of both the genome and proteins. Operations carried out on a ProGenGrid could include the aggregation, selection, retrieval, analysis, filtration and sharing of proteomic and genomic data for concurrent access and collaboration. The ProGenGrid, developed at the University of Lecce (Aloisio et al., 2005a), is intended to provide a practical solution to specific HealthGrid problems. This Grid aims at providing a virtual laboratory where e-scientists can simulate biological experiments, compose existing analysis and visualization tools, monitor their execution, store the intermediate and final output and finally save the model of the experiment for updating or reproducing it. Another study (Aloisio et al., 2005b) introduces the ProGenGrid workflow that comprises a semantic editor for discovering, selecting and composing bioinformatics tools available in a Grid environment, and a workflow scheduler for running the composed applications. The workflow editor uses an ontology of tools for the bioinformatics domain and employs the Unified Modeling Language (UML) for modelling the workflow. The UML graphical notation is stored as an XML file. On running the application, the workflow scheduler takes activities from the XML file and runs them, taking into account the state and availability of Grid resources and relevant bioinformatics tools. The system also allows monitoring of the job flows.

### **4.4.2 MediGrid**

The MediGrid (Medical Grid) is a type of HealthGrid designed specifically for accessing and sharing medical information around the globe by authorized individuals/organizations. It is expected to contain all levels of medical information from tissue, organ, and patient to population and public health, including various types of scans, mammograms, simulations and models of different body organs and other medical domains, etc. All this information needs to be shared and critically analysed for further medical research purposes. A paper proposes a MediGrid (Boccia et al., 2005) which has been designed specifically

for the aggregation and integration, analysis and visualization, and processing and management of biomedical images for nuclear doctors. It is a distributed, user friendly GUI-based application that uses the First In First Out (FIFO) algorithm for job scheduling and follows the Grid Application Development Software (GrADs) Project workflow (Berman et al., 2001), (Vadhiyar and Dongarra, 2005). It focuses on complex Grid-enabling parallel algorithms for the examination of medical images.

Amongst others, the MediGrid can be sub-categorized in terms of its practical application; representative examples include the MammoGrid, the NeuroGrid and the SimGrid, which are next described.

#### *a. MammoGrid*

The MammoGrid (Mammography Grid) (MammoGrid, 2007) is one of the most important practical examples of a MediGrid designed particularly for the access, storage, retrieval, analysis, management, manipulation and sharing of various types of digital images, medical scans, or DICOM (Digital Imaging and Communications in Medicine) files. Some computationally intensive image analysis algorithms often devised to assist clinicians to make decisions in diagnosis and therapy are known to produce better results, but are not used in practice due to the lack of computing power (Breton et al., 2005). The MammoGrid is expected to provide enormous computing and storage resources so as to make feasible and to support distributed image analysis.

A recent study (Scheres et al., 2005) presents an interface between Grid computing middleware and a three-dimensional electron microscopy (3D-EM) image processing package ("Xmipp") (Sorzano et al., 2004). Results showed clearly that 3D-EM image processing can greatly benefit from the resources offered by Grid computing. Another study (Glatard et al., 2005) produced a generic, Grid-enabled workflow framework, to be deployed on the computational Grid infrastructure provided by the EGEE European project (EGEE, 2007). It

encompasses image registration algorithms wrapped in standard Web-Services, a Grid enabled workflow manager, and Grid middleware for performing the distributed computations. The framework developed could easily be adapted to a wide variety of medical applications. However, one of the limitations stems from the *stateless nature* of Web Services.

e-Diamond, a UK e-Science project (Brady et al., 2003), is a Grid-enabled prototype system (medical image database) that aims at supporting breast cancer screening by maintaining a national database for digital mammograms. In the development of e-Diamond, an object-relational approach to the storage of DICOM files has been taken (Power et al., 2004). Other work carried out within the context of the e-Diamond research project (Power et al., 2005), (Simpson et al., 2005) addresses the challenges of patients' data security and confidentiality via employing query modification. Query modification is also used in GIMI (Simpson et al., 2005) to restrict access to the data in Grid-enabled medical research databases for the sake of patients' data security.

A MIP-Grid (Grid-enabled Medical Image Processing Application System) is presented (Huang et al., 2006), that is based on OGSA-DAI middleware. It aims at providing high performance medical image process services in a large distributed grid computing environment. OGSA-DAI allows uniform access to and integration of data held in heterogeneous data resources.

#### ***b. NeuroGrid***

The NeuroGrid (Neurology Grid) is another example of a MediGrid that is designed to support neurologists worldwide in their collaborative work. A recent study (Geddes et al., 2005) has proposed the implementation of a NeuroGrid, i.e. a Grid dedicated to neuro-scientific studies. It is intended to be built on the experience of other UK e-science projects aiming to assemble a Grid infrastructure, and apply this to three exemplar areas: (a) stroke, (b) dementia and psychosis, and (c) generic collaborative neuroscience research. Grid-enabled

sharing of data, experience and expertise will facilitate the archiving, curation, retrieval and analysis of imaging data from multiple sites and enable large-scale clinical studies in neurology. To achieve this goal, the NeuroGrid seems to be built upon existing Grid technologies and tools (developed within the UK e-Science programme), aiming to integrate image acquisition, storage and analysis, and to support collaborative working within and between neuro-imaging medical centres. Moreover, the Biomedical Information Research Network (BIRN) (Ellisman and Peltier, 2004) is devoted to neurology and is exploring the use of Virtual Data Grid (VDG) to support multiscale brain mapping. BIRN currently participates in three testbed projects; namely Function BIRN, Morphometry BIRN and Mouse BIRN (Stewart, 2004).

In the not too distant future, a dedicated NeuroGrid will address the need to support the computation and monitoring of various neurological functions, for both humans and animals, such as brain histology, MRI (Magnetic Resonance Imaging), neurological disorders, electron microscopy and brain imaging, etc.

### *c. SimGrid*

The SimGrid (Simulation Grid) is also an example of MediGrid designed specifically for providing special simulation and modelling services for various types of medical treatments and analysis such as surgery, radiotherapy, chemotherapy, endoscopy, electrocardiography, osteotomy and bio-fluids simulation, etc. Thus SimGrid encapsulates all simulation levels from Proteomics and Genomics up to overall body-level simulation. The SimGrid can be of importance not only in planning surgeries but also in training surgeons (Breton et al., 2005). The simulation process is quite time consuming and might require millions (or even billions) of computation cycles and terabytes of storage space, depending upon the nature of the specific simulation task. However, using Grids for this purpose could resolve the problem of computation speed to a considerable extent.

A recent study (Gonzalez-Velez and Gonzalez-Velez, 2005) presents a stochastic simulation of L-type  $\text{Ca}^{2+}$  current assuming thousands of calcium channels on the membrane of a spherical cell. The simulation runs on a dedicated Grid and employs structured parallelism techniques. Results showed hours of time saved using a computational Grid for simulation (compared to single-machine simulation runs).

GEMSS (Grid-enabled Medical Simulation Services) (Jones et al. 2004), (Benkner et al. 2005) that is concerned with the Grid-provision of advanced medical simulation applications and aims to provide a transparently accessible health computing resource suited to solving problems of large magnitude. The viability of this approach is currently being evaluated through six diverse medical applications, including maxillo-facial surgery planning, neuro-surgery support, medical image reconstruction, radiosurgery planning and fluid simulation of the airways and cardiovascular system. Without using a Grid, an accurate nonlinear simulation takes a considerably longer time (up to several hours), whereas, by allowing access to high performance computing through the Grid, the simulation time can be reduced to a level acceptable for clinical implementation (less than one hour), with the potential to improve the outcome of the surgical procedure. The GEMSS Grid infrastructure is based on standard Web Services technology with an anticipated future transition path towards the OGSA (Foster et al. 2003) standard proposed by the Global Grid Forum.

A new execution and simulation procedure for two dental applications, namely Computational Fluid Dynamics (CFD) and Computational Aero Acoustics (CAA) is proposed in (Nozaki et al., 2005), which can reduce the implementation time via Grid-enabled parallel processing. The study also reports on the design, implementation and performance evaluation of the optimal CPU resource allocation based on the total computation time of the dental application, which combines CFD and CAA as a part of a DentGrid system. The data for both the simulations is obtained by Magnetic Resonance Imaging (MRI). This DentGrid system aims to be a computation and storage power supplier for dental clinics and

hospitals. Simulating dentistry operations is highly beneficial, in the sense that dentists can examine visually the post-effects of dental surgery.

The modelling of individuals is an ongoing research topic and involves the complete simulation of the human body, which is a computationally intensive task. In the field of modelling and simulation, Grid computing has the capability to accelerate the pace of the analysis/discovery process and to deliver the new results quickly and efficiently to the medical user community (Berti et al., 2003).

The application of OGSA-DAI in Simulation Grids has been discussed (Xing et al., 2006), to address the issues of integrating, controlling and accessing the different types of distributed data resources in the simulation. The databases in the simulation grid system supported the dynamic distribution of the data and model resources in the simulation environment.

OGSA-DAI has been used also as a middleware in the BioSimGrid project (Wu et al., 2004), that aims to exploit the Grid infrastructure to enable comparative analysis of the results of bio-molecular simulations.

#### **4.4.3 BioMedicalGrid**

The BioGrid and MediGrid merge into the BioMedicalGrid which encapsulates features of both the Bio and Medi Grids. The main challenge faced in biomedical informatics is the development and maintenance of an infrastructure for the storage, access, transfer and simulation of biomedical information and processes. Moreover, BiomedicalGrids must be able to produce, use and deploy knowledge as a basic element of advanced applications and to achieve this, they are mainly based on Knowledge Grids and Semantic Grids (Breton et al., 2005). BiomedicalGrids will thus provide a universally accessible platform for the sharing of ever-increasing biomedical data pertaining to all the levels of healthcare such as molecule, cell, tissue, organ, patient and public health, etc.



They are expected to provide interoperability and sharing/collaboration to both the Biological and Medical domains of healthcare.

Recent research (Tirado-Ramos et al., 2005) has used *on-line application monitoring* for improved computational resource selection and application optimization. A number of user-defined performance metrics within the European CrossGrid Project's G-PM tool (CrossGrid.org, 2004), (Stevens et al., 2004) have been used to run a blood flow simulation application (solver) based on the lattice Boltzmann method for fluid dynamics. Results showed that online monitoring gives a more accurate view of computational resource status than the regular resource information provided by standard information services to resource brokers. Moreover, on-line monitoring has good potential for optimizing biomedical applications for more efficient computational runs. Other work (Alonso et al., 2005), (Tirado-Ramos et al., 2005) has shown how a BioMedicalGrid can enhance the processing of a biomedical application as well as the respective image analysis. The integration of a bio-physical model into a clinical augmented reality system is another challenging task, where Grid technology could be the key (Breton et al., 2005).

#### 4.4.4 PharmaGrid

Another important type of HealthGrid is the PharmaGrid (Pharmaceutical Grid), which focuses on the management and sharing of drug-related data to support operations such as clinical trials, dose computation, drug discovery, drug development, drug interactions, pathology and genomics, etc. that could be carried out in a collaborative environment to advance the quality of healthcare.

The pharmaceutical industry is a distinct domain with specific operations and processes, and is currently faced with many challenges. There is an increasing need for more innovative products that can target more effectively today's critical diseases. At the same time, there is a growing pressure for personalised medication (by using both phenotype and genotype), a move which will increase both the effectiveness and safety of medicines, but which will eventually shrink

the scale of economy and create a much more fragmented market. Furthermore, the data produced by the pharmaceutical industry is of the order of terabytes or petabytes in size and needs massive storage capacity. Moreover, various operations to do with the manufacturing of drugs and the dissemination of drug information need huge numbers of computational cycles. The results obtained from various drug experiments and clinical trials are of crucial importance and need to be delivered in a consistent and timely manner to the healthcare professionals and patients.

One of the key challenges of the pharmaceutical sector today is to manage, share and understand the medicines information in a way that facilitates and accelerates the Research & Development process. This progress suffers from poor information management due to inflexible, closed, heterogeneous, unconnected and segregated sources of information. It has now been widely recognised that Grid technology holds out the promise for a more effective means of sharing and managing information and enhancing knowledge-based processes in the Pharma R&D environment. The emerging PharmaGrid is a powerful new technology set to revolutionise the way medicines-related (“Pharma”) information is used. The PharmaGrid has the potential to address the “information” problem, with many benefits for the industry, in terms of boosting innovation in drug discovery, shaping clinical trials, reducing time to market, and reducing costs. Furthermore, Grid technologies have the potential to provide transparency and integration of information, break communication barriers, enhance communication and collaboration between the various actors (industry, regulators, healthcare and insurance providers, doctors and patients), and as a result to accelerate a large number of healthcare processes to do with pharmaceutical therapies.

Other benefits from the use of PharmaGrids include (Houghton, 2002):

- substitution of *in silico* for *in vitro* and *in vivo* testing;
- operation and management of clinical trials;
- monitoring post-launch usage and outcomes;

- marketing and distribution of medicines;
- e-commerce and total quality management in healthcare supplies and procurement;
- regulatory and watchdog activities;
- financial planning and cost efficiency in healthcare;
- health information services for all stakeholders;
- electronic prescription and clinical decision support tools.

Even more crucially, the Pharma industry and researchers are faced with a continuously growing amount of distributed heterogeneous information, a real explosion of experimental data, documents, article, patents, with rapidly changing terminology and analysis approaches. In order to adequately fulfil such requirements, the PharmaGrids have to meet the following challenges:

- Intelligent middleware that facilitates the user transparent access to many services and execution tasks
- High quality security features, enabling large databases to be accessed via Grid solutions
- Sophisticated semantic and contextual systems to enable diverse sources of data to be related to knowledge discovery

Thus PharmaGrids are expected to deal with all types of drug-related information such as drug features, design specifications, safety, success rate, purpose and usage, and complex operations such as clinical trials, evaluation process, experimental results, treatment, effective trails, etc. This information should be shared across various organizational boundaries and manipulated online.

The development of PharmaGrids is instrumental in meeting the current industry challenges, as it will provide an efficient way of exchanging and managing knowledge in a shared environment in the areas of discovery, development, manufacturing, marketing and sales of new drug therapies. Grid infrastructures are

currently built upon different architectures, designs, technologies, open standards, and operating systems. PharmaGrid development is a highly complex and technically challenging activity and it should address many different problems to do with Pharma information, including knowledge-representation and integration, distributed architectures, search and access controls, data mining and knowledge management, real-time modelling and simulations, algorithm development and computational complexity. PharmaGrids will need to be scale-independent/scalable, adaptive, secure and dependable Grid infrastructures that enable the management of large networked distributed resources across different platforms of stakeholders, such as pharmaceutical companies, policy makers, R&D development companies, etc. The required enabling technologies include amongst others semantic web and agent-mediated approaches, peer-to-peer technologies and self-organising architectures.

PharmaGrids can be part of or closely integrated with other HealthGrids. For reasons of competitiveness and intellectual property protection, PharmaGrids are predominantly private, enterprise IntraGrids with strict access and authentication controls, but there is a recognised need for cross-industry platforms (InterGrids), whereby the resulting integration will lead to more efficient coordination of activities.

Moreover, PharmaGrids open up the perspective of cheaper and faster drug development and may enable parallel processes in drug development, away from the traditional approach where the full cycle of target discovery, target validation, lead discovery, lead optimization and transition to development takes on average 12 years (Breton et al., 2005). PharmaGrids hold the promise to provide improved and efficient drug design and better control of diseases and to improve *patient safety* and *quality of healthcare*. Examples of PharmaGrids could include dedicated Grids for Drug Discovery, Drug Modification, Management and/or Running of Clinical Trials, etc.

Another study (Tohsato et al., 2005), uses Globus Toolkit 3 and OGSA-DAI, for the federation of heterogeneous databases, for supporting a drug discovery process. Due to the rapid progress of biotechnology, there are an increasing number of life science databases, which need to be shared to conduct research collaboration and OGSA-DAI could make this feasible.

#### 4.4.5 CareGrid

The CareGrid is designed specifically for the general public healthcare services such as patient-centred or virtual healthcare services, dose computation, self-assessment, online health management, etc. Services on the CareGrid could be customized according to the individual patient needs so as to provide personalized healthcare services. Moreover, the CareGrid aims to provide data management facilities and improved diagnosis.

A typical example of CareGrid is a recently implemented prototype for HealthInfoGrid (Bilykh et al., 2003). The HealthInfoGrid can also be viewed as a Service Grid and its services are designed for sharing and distributing medical information, at times of critical importance and under strict privacy and security regulations. Formalization of the interaction semantics of the HealthInfoGrid components is based on coloured Petri-nets (Jensen 1997). HealthInfoGrid has various components such as organization, staging area, initiator, translator, and merger/adder.

CareGrids represent a new facet of advanced and improved healthcare that can provide personalized healthcare services at a cost-effective price. Patients can access the CareGrid to retrieve information about their own health, such as clinical tests and diagnosis, dose composition and recommendation, precautions and preventions, etc. This would save them the time and effort spent in waiting to book appointments and all the hassle that goes with such processes.

Examples of CareGrids could include dedicated Grids for Primary Care, Social Care, Home Care, Chronic Care, etc.

## 4.5 Types of Resources on HealthGrids

HealthGrids vie to bring the e-Health concept into reality, by helping to provide personalized and quality healthcare at a cost-effective price. In addition to possessing characteristics of generic Grid resources, the HealthGrid resources bear specialized features, which are adapted to the needs of healthcare domain. For example, data resources can include specialized health-specific data such as patient records (patients' complete health history, etc.), doctors' data (doctor's expertise, doctor's patients treated or under treatment, etc.), data files containing simulation results, digital images or DICOM files (mammograms etc.), medical communications, data related to pharmaceutical companies (drug design, effectiveness trials, etc.), medical instruments and other medicines information, etc.

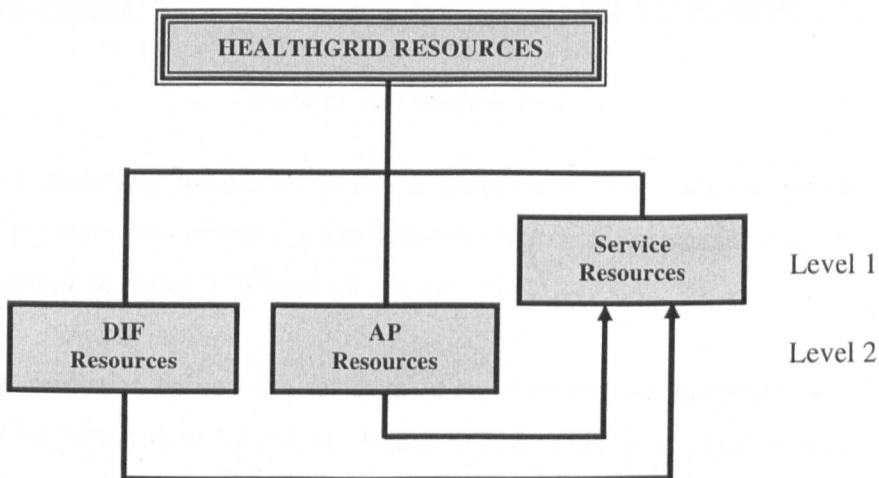
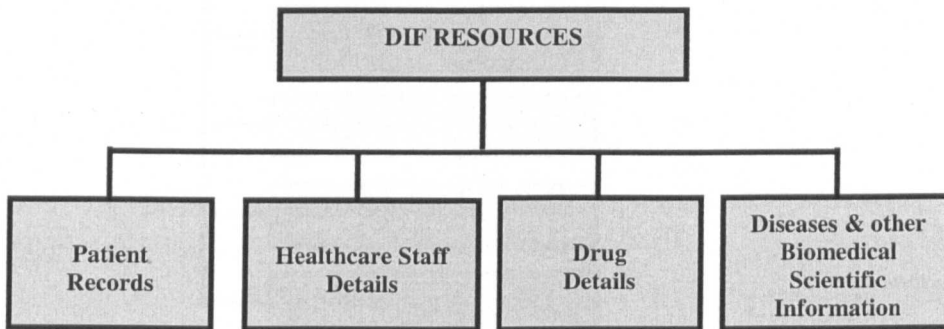


Figure 10: A Hierarchy of HealthGrid Resources

The specialized HealthGrid resources can be Data or Information or Files (DIF), Applications & Peripherals (AP), and Services. With a view to offer a unified taxonomy of specialised HealthGrid resources, the following sections propose and discuss a new hierarchy (see Figure 10).

#### 4.5.1 Data, Information or File (DIF) Resources

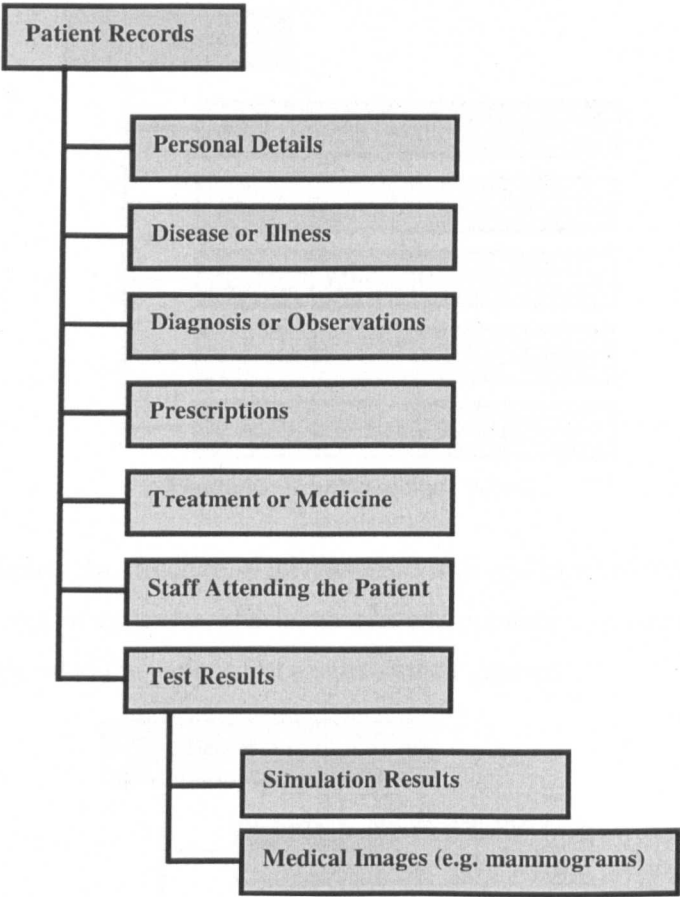
Information can be extracted from data which resides in computer files, therefore data, information or files are all considered as a single type of HealthGrid resource. This kind of resource comprises all sorts of medical records and healthcare information such as Patient Records; Healthcare Staff Details; Drug (medication) Details; and Diseases & Other Biomedical/Scientific Information.



**Figure 11: DIF Resource Tree**

Figure 11 depicts the breakdown of DIF resources in the form of a tree structure. Each of the end nodes of this tree can be further broken down into sub-categories, such as shown in Figure 12-Figure 15, respectively.

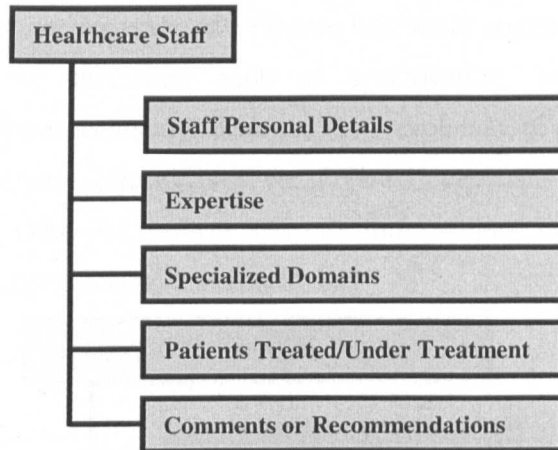
The DIF resources encapsulate all sorts of medical and health-related data or information which is contained in electronic files, such as medical images or mammograms, simulation results, clinical trials, radiotherapy reports, drug effects, patient and healthcare staff details, etc.



**Figure 12: Patient Records Details**

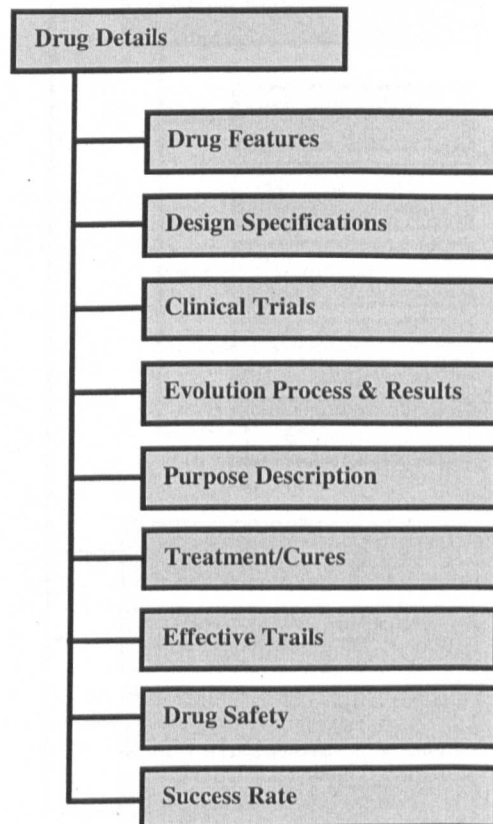
Figure 12 depicts the structure of a patient record. It includes the patient’s personal details and complete health (medical) history, the time he/she first needed a health treatment, the illness or disease diagnosed, healthcare staff that attended the patient, observations made about patient’s health, the type of treatment or medicines prescribed, various tests and their results such as mammograms or simulation results, etc. In fact, the patient record is the most sensitive data resource and is of crucial importance since it encompasses many critical elements related to personalized healthcare. A study in (Whiddett et al., 2006) suggests that patient’s information could be distributed with their consultation. Mostly, patients are happy to consider sharing their information with health professionals if they are first consulted.





**Figure 13: Healthcare Staff Details**

Figure 13 depicts the structure of a Healthcare staff record which contains details about their area of expertise, specializations and comments or recommendations made with regard to the various test results of their patients.



**Figure 14: Drug Details**

Figure 14 depicts the structure of a drug record which contains details about the various drugs or medicines such as specification, patient information, manufacturers/Pharmaceutical industries, drug evolution or drug development, clinical trials, purpose, effectiveness, etc. Figure 15 depicts a detailed breakdown of “Diseases & Other Biomedical Scientific Information” such as their nature, level of occurrence, specialized treatment, etc.

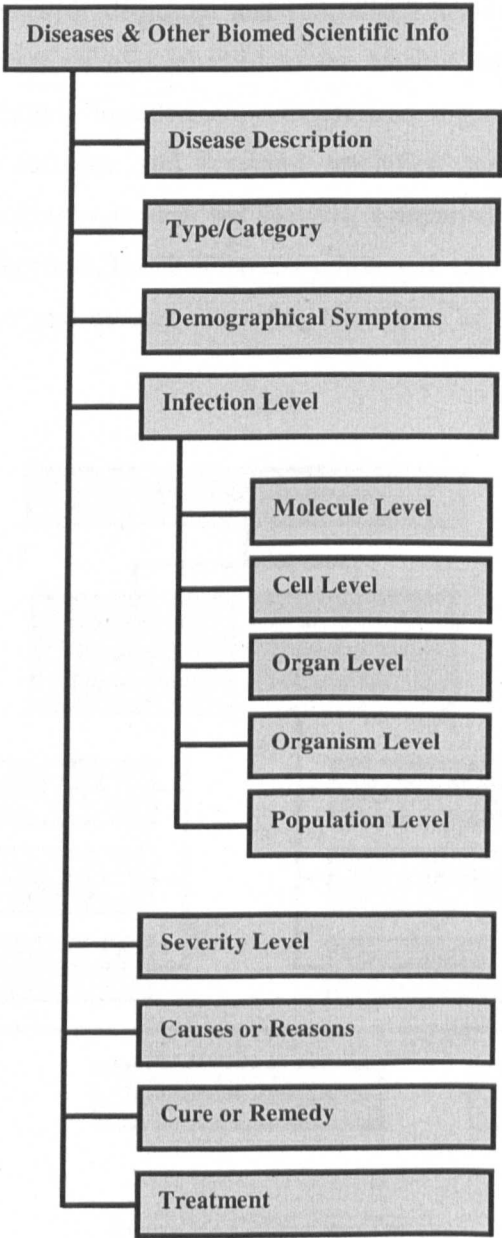


Figure 15: Diseases & Other Biomed Scientific Information

### 4.5.2 Application & Peripheral (AP) Resources

Applications such as device drivers or programs are installed onto computerized peripheral devices or machines such as image scanners, etc. Since software applications need hardware to work and cannot run alone, therefore both applications and peripherals are considered as a single HealthGrid resource. Such a resource comprises of dedicated and specialized healthcare applications and automated peripheral devices attached to the HealthGrid, such as image-scan processing applications installed on computerized organ scanners and image processors, MRI software and scanning machines, magnetic strip scanners (immuno-chromatography reader) for analysis, comparison and management of images, pattern classifiers, PACS (Picture Archiving & Communication Systems), simulation software and devices supporting simulation of body parts for various purposes.

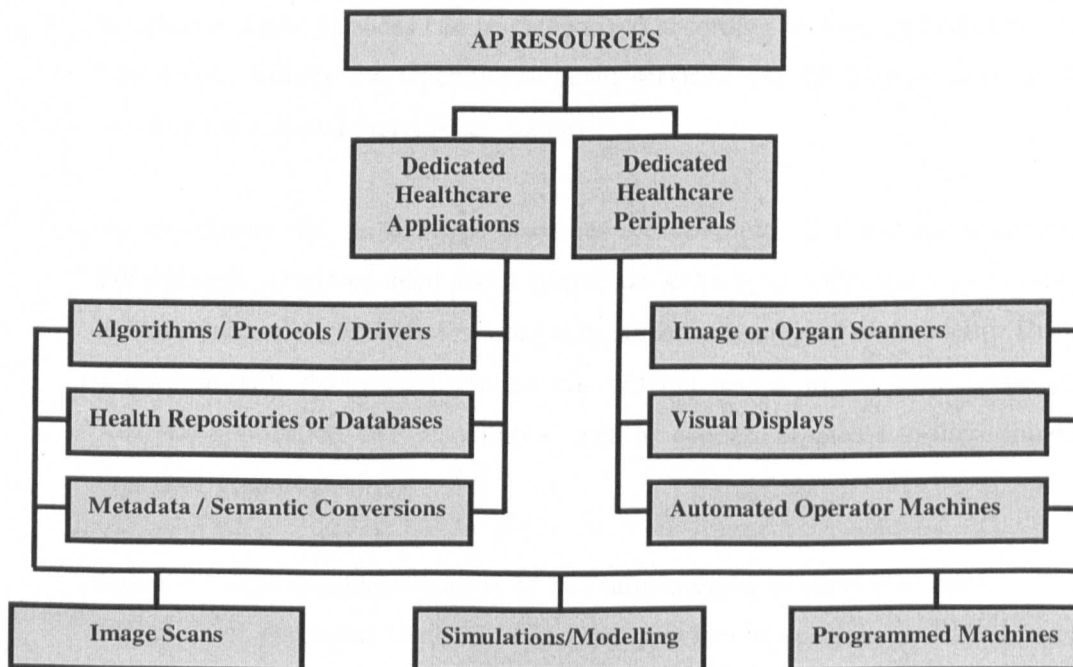


Figure 16: AP Resource Tree

Figure 16 depicts the breakdown of Application and Peripheral resources dedicated to healthcare. These two subcategories merge at the bottom of the tree to provide automated and/or programmed systems that can carry out various computational health-related tasks/operations. These resources could be managed through a dedicated HealthGrid, such as an *InstruGrid* (Instrument Grid), which could be designed specifically to access, retrieve and manage the scientific instruments used for the medical purposes. These instruments encompass the logical (software application) and physical (hardware machinery or peripheral) resources of the HealthGrid, in order to carry out specialized medical processes or tasks.

### 4.5.3 Service Resources

Service resources (also referred to as service-type resources or simply *services*) on a HealthGrid are compositions of basic healthcare-specific services, which are built using “standard” Grid services to provide high-quality customized healthcare. These services can be categorized according to their applicability into two levels: namely the Operational Level services and the Management Level services (Naseer and Stergioulas, 2006a).

Almost always, the service-type resources access/employ DIF and AP resources. For example, a Grid-enabled data Comparison & Analysis application can be used to carry out analytical studies of images or scans from any part of the world. Thus, a purpose-built data-access service can provide access to the Comparison & Analysis application (AP resource) as well as concurrent access to these image files (DIF resource).

Figure 17 depicts a classification of the various types of service resources. The first class of dedicated HealthGrid services provides *concurrent and ubiquitous access to or retrieval of* various other HealthGrid resources such as data, files, information, applications and peripherals, etc.

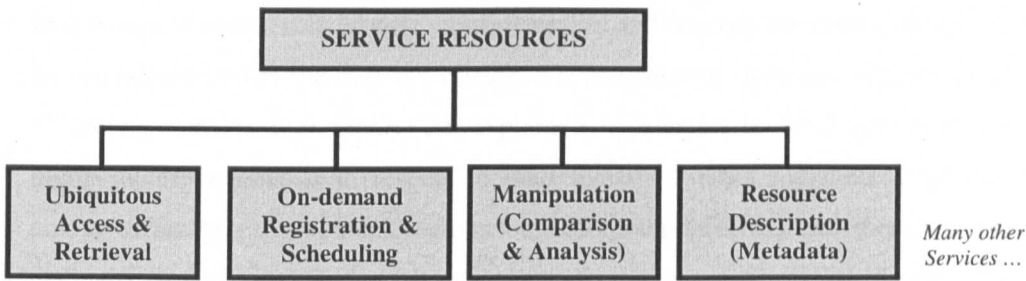


Figure 17: HealthGrid Service Resource Tree

The *on-demand registration & scheduling* services are responsible: (a) for authenticating the user registration and user authorization, groups or companies to access and use HealthGrid resources; and (b) for job scheduling and prioritizing healthcare processing tasks. The third class of services is responsible for performing *manipulation* healthcare operations (*e.g. data integration, analysis, comparison and reviewing*) by recruiting other HealthGrid resources. Some service-type resources are responsible for supporting online group-based data sharing in different parts of the world, a task made feasible by the so-called *resource defining services* (using metadata or semantic descriptions of the resources). There is a diverse range of many other services (Naseer and Stergioulas, 2006a), which can be provided on HealthGrids.

## 4.5 Resource Discovery in HealthGrids

### 4.5.1 The Resource Discovery Challenge

The successful adoption and integration of Grid technology in the healthcare industry has not yet been achieved fully due to reasons such as awareness, reliability, trust, and security. Probably these are also some of the reasons why the healthcare community is reluctant to embrace an automation revolution as witnessed in other industries (*e.g. banking, finance, marketing, stock exchange, etc.*). People would prefer to be operated on by a human surgeon rather than being laid down helplessly in front of an automated surgery machine (surgical robot).

In the case of health-related data, confidentiality and security are crucial factors to be considered before the data is exchanged across various domains, organizations or among various individuals. As explained in Section 0, Grid resources are highly heterogeneous with respect to their dynamic status and their dispersion across different geographical regions or different platforms, and therefore are difficult to manage. In an interconnected environment of healthcare, the successful implementation of *resource discovery* techniques need to cope with issues such as scalability, autonomy and heterogeneity, security and privacy, maintainability and evolvability, reliability and robustness, active and autonomous coordination, mobility and ubiquity, etc (Bilykh et al., 2003). Moreover, differences in medical terminologies, organizational heterogeneity and ubiquitous management of resources can hinder consistent and accurate resource discovery on HealthGrids.

#### **4.5.2 Heterogeneity and Coding Issues**

One of the most critical issues when developing a medical database is the provision of appropriate mechanisms for allowing updates and tracking changes. This importance is derived from the legal and ethical requirements to record all updates of patient and screening data (Power et al., 2004). Also, the many geographically distributed healthcare organizations have data stored in different formats, different database management systems, for different types of medical tests, and in different descriptions. Even the medical terminologies used across various organization boundaries vary widely, thus making resource discovery a highly complex process, since there is a possibility that the information is misinterpreted by the user (individual or organization), which may lead to serious medical errors. Moreover, the remote processing of a dataset is also an issue as regards to obtaining consistent (error-free) results every time.

Various healthcare terminologies and classification systems (Coiera, 2003) have been developed such as the International Classification Diseases (ICD-10), the Systematized Nomenclature of Medicine (SNOMED), the Medical Subject

Headings (MeSH), the Unified Medical Language System (UMLS) and the Medical Dictionary for Regulatory Activities (MedDRA), etc.

The UMLS is the latest system with the potential of wide acceptance, since it provides a mapping between 100 different terminology systems and incorporates a number of other systems such as ICD-9, ICD-10, MeSH, ICPC-93, WHO Adverse Drug Reaction Terminology, SNOMED II, SNOMED III and UK Clinical Terms. The knowledge sources of UMLS are the metathesaurus, the semantic network and the lexicon. UMLS is used in (Slaughter et al., 2006) to identify the underlying semantics of health consumers' questions and physicians' answers in order to analyse the semantic patterns within their texts. Semantic relationships are manually identified within the question-answer pairs from Ask-the-Doctor Web sites. Identification of the semantic relationship instances within the texts is based on the relationship classes and structure of the UMLS Semantic Network since its relationship classes are hierarchical.

None of the healthcare terminologies and classification systems has so far succeeded in resolving the issue of heterogeneity in medical terminologies. However, the coding systems should be compared on specific tasks and results should cautiously be generalized to other tasks and populations. Similarly, the poor performance of the coding systems on tasks outside their design should not reflect badly on their assessment in terms of performance capability (which should be always assessed within the intended scope of application). Therefore there is a well-recognised need to have a flexible and detailed medical terminology and classification system to integrate medical or health-related data over the HealthGrids, which are suitable channels for ubiquitous access to, sharing of, and processing of masses of health-related data. There are many other technical issues which make resource discovery a major challenge and need to be addressed for the successful implementation of HealthGrids.

### 4.5.3 Resource Discovery Problems and Solutions

Resource Discovery in HealthGrids is an important and timely issue that relates to a number of “information integration” problems in health informatics. The integration of medical information electronically on a national and international level in a way conformant to all the organizational policies and Grid constraints is a big challenge. Moreover, coding & terminology are far from unified across the sector, and medical coding systems are not ready to incorporate and manage the emerging *genetic information* (Breton et al., 2005). Thus it is important to maintain standardized metadata and standardized translation between the various medical terminologies used in different countries.

Emerging Grid technologies combine Web Services with Grid infrastructures and employ Semantic Web technologies for solving problems of Grid deployment and management. Such *semantic description schemes* can provide the glue for coordinating different sorts of resources, services and application on a HealthGrid and between HealthGrids (Stevens et al., 2004), (MammoGrid, 2007).

There is also a need to have *enhanced investigation mechanisms* in healthcare information systems so as to facilitate the process of universal automation and ubiquitous sharing and to achieve successful resource discovery on HealthGrids.

Each of the HealthGrid resources described in Section 5 has its own discovery problems, some of which are discussed in the Table 2.



Table 2: Resource Discovery Problems and Possible Solutions

<i>Resource Type</i>	<i>Sources of Problems in Resource Discovery</i>	<i>Possible Solutions</i>
DIF	Encoding medical terms Compatibility of file formats Matching ontologies Semantic descriptions Data/information consistency Ubiquitous & easy availability Heterogeneous Databases/Languages Data archiving & distributed image analysis	Devising and using interoperable standards for healthcare terminologies and classification systems. For example, this can be based on any of the following systems: SNOMEDor UMLS (Coiera, 2003), or MedDRA (Medical Dictionary for Regulatory Activities, 2007). Also designing generic protocols for ubiquitous and easy availability of data and standardization of data formats. In a similar vein of work, the HL7 community (Health Level Seven Inc., 2007), (Dolin et al., 2001) is creating standards for the exchange, management and integration of electronic healthcare information.
AP	Reliability Compatibility <ul style="list-style-type: none"> <li>• inter &amp; intra applications</li> <li>• inter &amp; intra peripherals</li> <li>• inter &amp; intra Grid infrastructures</li> </ul> Grid-enabled applications & peripherals Consistency	Designing dedicated Grid-enabled applications and peripherals having platform and organizational compatibility. Also compatibility with inter- and intra- Grid infrastructures to support ubiquitous performance and exchange of information.
SERVICE	Accuracy Integrity Cost effectiveness User friendliness Efficiency and performance Security and user authentication [Privacy, Integrity, Confidentiality]	Designing specialized protocols to enable controlled access and performance monitoring to ensure high performance at a cost-effective price. Also providing easy to use flexible IDE (integrated development environment) for dedicated HealthGrid interfaces. Moreover, implementing security measures for user privacy and confidentiality.

## 4.6 Conclusion

HealthGrids can be used to support many kinds of healthcare operations and tasks. The case for the use of Grid technology in healthcare arises mainly from the need to improve, safeguard and effectively exploit the available *life-significant medical information*, the need to protect the privacy of *personal, life-sensitive health information*, and the need to provide *integrated healthcare services* and have in place effective, global *channels of collaboration*. To do all this effectively, different types (the most suitable ones) of HealthGrids should be employed to perform different dedicated tasks with specialized features and functionalities. For the long-term future, there is a need for the various Grid-enabled applications to be designed specifically for HealthGrids.

This chapter has reviewed current implementations of HealthGrids, and offered a systematic taxonomy of the HealthGrids and their resources. It has outlined the characteristic features and functionalities of HealthGrids and reflected on the need for Grid technology in healthcare. Based on their functionality, purpose, and application area, a taxonomy of HealthGrids has been proposed into four major types. It has been shown that each serves a dedicated purpose, so as to provide the required services and to support the performance monitoring of the specified/desired tasks. Furthermore, the types of HealthGrids have been examined on an individual basis and their representative implementations have been reviewed.

A unified taxonomy of HealthGrid resources has been proposed and the issue of resource discovery in HealthGrids has also been discussed. A new refined hierarchy is presented, where specialized HealthGrid resources can be categorised into three major types; namely, Data or Information or Files (DIF); Applications & Peripherals (AP); and Services. In summary, the chapter has considered the challenge of resource discovery; discussed the problem of heterogeneity, issues of medical coding and terminology, and the role of semantic technologies; and proposed potential solutions for different types of resources.

To exploit effectively the wealth of medical information, there is an urgent need to integrate, manipulate, process, and analyse huge heterogeneous datasets from disparate sources. More systematic use of Grid technology in healthcare will not only help meet the current needs for data processing, but will ensure that future demand for even more capacity to deal with far larger volumes of data can be met.

Data security is another major issue, as healthcare data has to be protected through ethical firewalls to ensure the privacy, confidentiality and integrity of patients' data. HealthGrids deal with 'life sensitive data' and the patient record is the most sensitive data resource that encompasses many critical elements related to personalized healthcare. As patient record encapsulates instances of various other entities, HealthGrids can help achieve the required levels of robustness and consistency of the patient record. HealthGrids can also serve as an effective channel for international collaborations.

To address the challenge of resource discovery in HealthGrids, a systematic search strategy should be devised and adopted, as the discovered resource should be valid, refined and relevant to the query. Standards should be implemented on domain-specific metadata. Moreover, a critical question arises as to how metadata can support the integration of two or more heterogeneous objects. There is also a need to have a semantic integration of various resources that are geographically or organisationally spread, so that they can be shared and utilized globally on a HealthGrid. The emergent semantic networks ensure the integrity of meaning between different concepts and can play an important role in solving this complex integration problem. Moreover, it has been witnessed from the literature survey that the mainstream Grid technologies such as OGSA-DAI can prove to be a candidate solution to the data federation problem.

As a final remark, HealthGrids offer accurate and reliable sources of health information that can be accessed any time from any place. They can and should become a major driver in the race towards successful e-Health and an important ingredient of the next generation of healthcare IT. A successfully implemented

HealthGrid infrastructure could support all the facets of healthcare sector, and help realise the vision of personalized healthcare. Successful implementation of HealthGrids will have a high impact towards lower costs and greater benefits for healthcare in the long run.

## **PART-II**

## Chapter 5

### Research Methodology

#### 5.1 Introduction

The research method adopted in this study is the Empirical Research Method (Johnson, 2003). It is used for verifying the hypothesis that is based on the observations and contextual analysis derived from the literature surveys in the previous chapters. In order to investigate the problem of *“how to facilitate the semantic federation of heterogeneous data resources using mainstream Grid technologies”*, a set of current Grid technologies were explored during the technology analysis phase that lead to the design of ASIDS, an n-tier-to-n-tier application Architecture for Semantic Integration of Data Sources, which was later implemented within an exemplary ASIDSApplication environment (the HealthGrid exemplar) in Chapter 7, in order to test the hypothesis of this thesis. This chapter discusses the adopted research methodology, various phases, and its suitability for the study.

#### 5.2 Research Approach

The Empirical Research Method has been used as this research involves implementation of the proposed architecture within an exemplary ASIDSApplication environment (the HealthGrid exemplar). This prototype would be built to experimentally investigate the hypothesis of this thesis, which is formulated as: *Existing mainstream Grid technologies are sufficient for providing effective and sustainable solutions to the problem of semantically federating networked (heterogeneous) data resources.*

In computer science, the Empirical Research Method generally follows four distinct steps: hypothesis generation stage – where the hypothesis to be investigated is formulated, method identification stage – where the method or technique used to examine the hypothesis is identified (generally it includes experimentation), result compilation stage – where the results of the experiment (prototype implementation) are evaluated and finally the conclusion stage – where, on the basis of the evaluation, the conclusions are drawn and the hypothesis is proved or disproved (Johnson, 2003).

In the method identification stage, the Rapid Prototyping Model was followed (Tripp and Bichelmeyer, 1990) as the software engineering process model for the development of an experimental prototype. The Rapid Prototyping Model is chosen, as it is a viable and appropriate model for the instructional design, especially for computer-based instruction, as one can get an idea of the final model beforehand and any problems, issues (related to the model design) are highlighted before the model is finalised. The Rapid Prototyping process is depicted in Figure 18 (Research design) as one of the phases of the research design.

### 5.3 Research Design

This research aims at exploring the possibility of using the mainstream Grid technologies to semantically integrate heterogeneous data sources in an efficient, sustainable, and user-friendly way. As seen in previous chapters, semantic interoperability of geographically distributed and heterogeneous data resources is a critical issue, and Grid technologies have the potential to address this issue in a systematic manner, making applications more scalable and manageable. One of the main contributions of this research is proposing a simple solution to the very complex problem of data resources integration and interoperability in Grids.

The research has been carried out in seven different phases which are depicted in Figure 18.

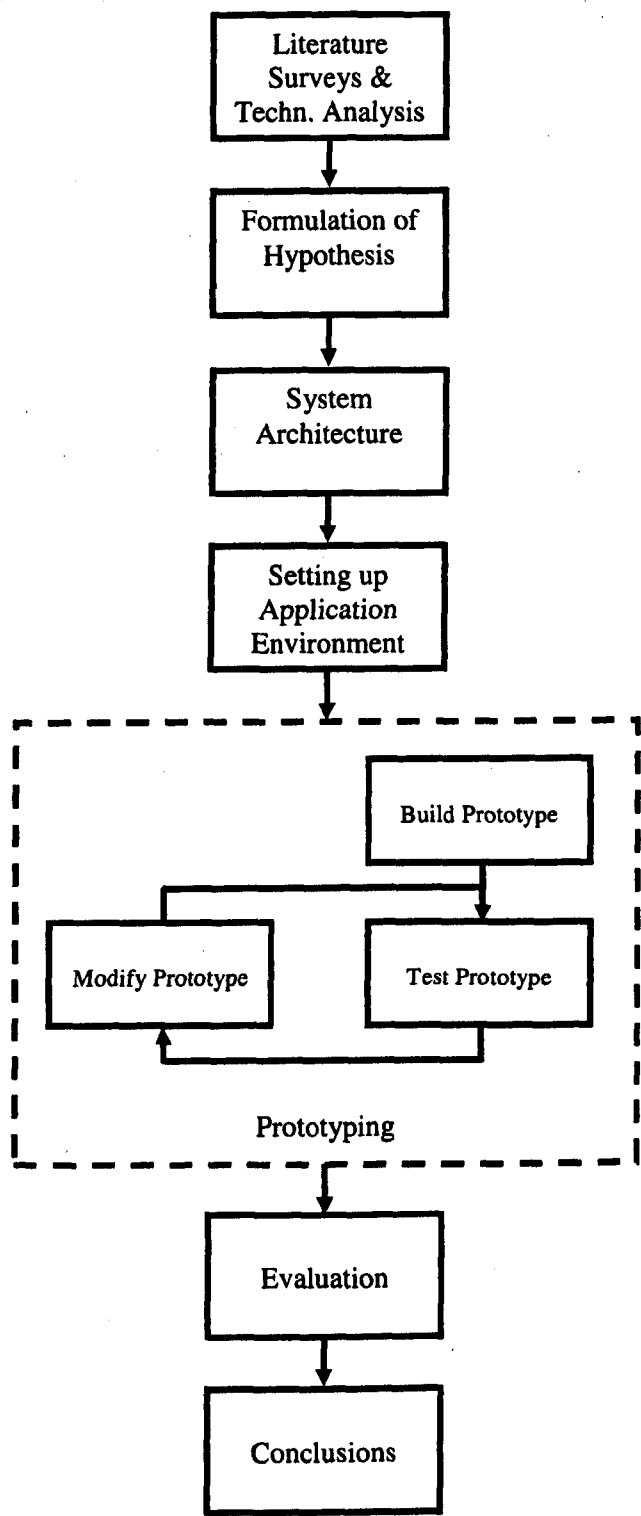


Figure 18: Research Design



### 5.3.1 Phase I: Literature Survey

Grids have been a relatively hot topic in the past few years and expected by many to be quite promising in the way of 'resource virtualization' and integration. Resource sharing is one of the most important and outstanding features that Grid technology can provide in order to facilitate the globalization of heterogeneous resources.

Among many different types of Grid resources, as discussed in Chapter 3 (Figure 6), this research focuses mainly on the Data-type resources shown in Figure 11 (Chapter 4). Therefore, the mainstream Grid technologies are explored to assess their capability to semantically federate networked (heterogeneous) data resources.

This research initiated with the literature survey and contextual analysis of the related literature. Taking into account the potential of mainstream Grid technologies to solve complex computation and storage problems (Joseph et al., 2004), the allied literature, related to the resource discovery issues on both the Grids and HealthGrids, was explored and different resource discovery techniques were analysed, proposing taxonomies of resource discovery models and types of HealthGrids, respectively as a result. Also, a technology analysis was conducted in order to see what do the available mainstream Grid technologies offer in order to address the data integrity issues. These technologies were used in the proposed architecture. Moreover, in this phase the key problems were identified, highlighted and analysed. Through this analysis, the research question was devised and moulded into a hypothesis. The by-products of this phase were taxonomies described in earlier chapters.

### 5.3.2 Phase II: Hypothesis Formulation

Based on the contextual analysis carried out through comprehensive literature surveys, it was concluded that resource discovery is a major problem for Grids and other Distributed environments (Naseer and Stergioulas, 2006c). It is

important because there is a need (in research as well as in sectors of the economy) to dynamically integrate and share resources (both static and variable) globally, in an effective & user-friendly way and to solve complex resource-sharing problems. To answer the question of how to facilitate the semantic federation of heterogeneous data resources using mainstream Grid technologies, a hypothesis was formulated to see whether *Existing mainstream Grid technologies are sufficient for providing effective and sustainable solutions to the problem of semantically federating networked (heterogeneous) data resources.*

### 5.3.3 Phase III: System Architecture

Based on the conclusions drawn from the literature survey and the related technology analysis, a design for *ASIDS: an Architecture for Semantic Integration of Data Sources* is proposed. In order to validate the hypothesis of this thesis, the ASIDS architecture is later implemented in Chapter 7 in a form of a prototype within an exemplary ASIDSApplication environment (the HealthGrid exemplar), which was set up for this purpose.

The ASIDS architecture follows an n-tier (Hyatt, 2007) design model and constitutes of three main components. The three components of this architecture are the Physically Distributed (heterogeneous) Data Sources (PDDS), the Semantic Query Engine (SQE) and the Web-based User Interface (WUI). Each of these components is composed of (or distributed over) n tiers. Hence it is an n-tier-to-n-tier architecture design, where each component can be split into further n-tiers. By using this architecture, any multi-tier (constituting of more than one tiers), data integration application can be easily implemented over a Grid in a distributed manner. The architecture is particularly suited for achieving semantic interoperability among geographically distributed heterogeneous data resources. All the components of this architecture are loosely coupled in a distributed fashion.

### 5.3.4 Phase IV: Setting up an Exemplary Application Environment

The next phase was to set up an exemplary HealthGrid environment for the implementation of a prototype based on ASIDS. From the literature surveys and the related technology analysis, it was seen that the technologies which could be used to accomplish this research task included Globus Toolkit (GT4) and OGSA-DAI (Open Grid Service Architecture Data Integration and Access) with other Web Services technologies such as XML (Extensive Markup Language). Hence for this implementation, it was necessary to carry out installations of various mainstream Grid technologies such as GT4 and OGSA-DAI, and to construct heterogeneous data sources (experimental databases). The databases acted as the testbed for this pilot study (HealthGrid exemplar). Heterogeneity of all the data sources was ensured in terms of their heterogeneous fields. More specifically, the field labels of all the data sources were different but they contained the same (type of) information (i.e. record values). The application environment was set up according to the ASIDS architecture design, and the ASIDSApplication prototype was build on top of it.

This phase naturally overlaps with Phase V and was partially carried out in parallel to Phase V.

### 5.3.5 Phase V: Prototyping

While in Phase IV the HealthGrid exemplar environment was set up by carrying out the necessary installations and the data sources were successfully constructed, the experimental prototype, called the ASIDSApplication, was built using Grid technologies in order to test the proposed architecture and check the validity of the hypothesis. This prototyping was done in three sub-phases, namely: build, modify, and test (the last two being in an iterative loop), to attain consistency in the prototype design and functionality. The Core Grid services from the Globus container (Globus Toolkit version 4.0) were run, and OGSA-DAI (Open Grid Services Architecture Data Access and Integration) was used as an interface for accessing the heterogeneous data sources.

The ASIDSApplication consists of three main components: a JSP page (called DDQuery.jsp), a client Servlet (called DataDiscoveryClient.java) and a JAVA class for semantic mapping (called Mapping.java). The JSP page acted as the GUI interface and received queries from the users. The ASIDSApplication was built using the JAVA JDK1.5.0\_09 software development kit (Eclipse SDK 3.2.2), which ran on Apache Tomcat Server (version 5.0.28). Based on the user query, it then fetched data from various data resources regardless of their geographical locations, heterogeneous formats and semantics (on field-level) and made this data available on the (Health) Grid. The data retrieved was displayed on a webpage and could be further used to perform desirable operations such as scientific collaboration and group-wise or exploratory analysis in HealthGrids, eventually promoting e-Health (the entire Prototyping phase is explained in detail in Chapter 7).

### 5.3.6 Phase VI: Evaluation

In this phase, with the aim to test the implementation of the ASIDS architecture on the HealthGrid prototype and to demonstrate the feasibility of the proposed approach, two types of tests were conducted: (a) *Experiment-I* for testing out the system with single Grid Installation (GI) and large datasets, (b) *Experiment-II* for testing if the system works multiple Grid Installations (GIs) each having one Data Source. For this reason both the experiments were conducted in different network setups. The elapsed time measurements were taken and results were plotted on the graphs. Results showed that the proposed semantic integration approach (ASIDS architecture) remains functional in both the experiments. Moreover, it is expected that the architecture would still be manageable, reusable and flexible in case of even larger numbers of GIs and even increasing Data Sources just by making minor changes to the system configurations.

### 5.3.7 Phase VII: Conclusions

Finally, the last phase of this research was to draw conclusions from the entire study in order to seek the validation of the hypothesis based on the literature

reviews, the empirical work of this research (experimental prototype) and the results from the subsequent evaluation analysis. This study advances the state-of-the-art in the field by providing a simple, effective solution to the complex challenge of semantically integrating heterogeneous data sources and making them available on Grids, e.g. for facilitating collaborative research in the HealthGrid exemplar. Moreover, avenues for future research in this area are discussed.

## **5.4 Conclusion**

This chapter described the overall research methodology and the approach used to carry out this study. The seven-phase research design was explained in detail, going through each of its phases. The employed Empirical approach seems to be well suited to address the research question.

## Chapter 6

# ASIDS: Architecture for Semantic Integration of Data Sources

### 6.1 Introduction

Based on the investigation of the problem through literature survey and the related technology analysis, this chapter proposes a design for *ASIDS: an Architecture for Semantic Integration of Data Sources*, that is later used in Chapter 7 to build a prototype, in order to validate the hypothesis of this thesis, which states that: *Existing mainstream Grid technologies are sufficient for providing effective and sustainable solutions to the problem of semantically federating networked (heterogeneous) data resources.*

This chapter describes the proposed ASIDS architecture, explains its different components, and elaborates on the choices made in the selection of the various tools and technologies used. The functionality and effectiveness of the architecture is discussed and conclusions are drawn at the end.

### 6.2 Proposed N-Tier-to-N-Tier Application Architecture

In the review conducted in Chapter 3, it has been seen that the semi-distributed resource discovery model may provide the best option for creating resource/request brokering systems and designing related middleware packages, since overall it seems to be more reliable. Moreover, it was seen that using a Hybrid approach over a Semi-Distributed architectural model can help resolve the problem of resource discovery (even for data-type resources) in Grids to some extent. Taking into account the observations from the literature surveys, the proposed ASIDS architecture was designed. Since it follows the Semi-Distributed

architectural model, its design is expanded to more than one tiers (n-tiers). Moreover, using the Hybrid approach (by combining the P2P and Semantic approaches), in a sophisticated manner, as suggested in the literature survey of Chapter 3 can provide an “optimal” solution to the problem (under the current technology constraints):

N-tier architectures (Hyatt, 2007) are composed of layers or sections, each of which is a standalone entity which can communicate with layers above and below it. Each section is independently designed and protected from the others by creating extensible interfaces. Therefore, all changes made to a layer are usually encapsulated within the layer and if the change is not a major one, then it will not necessarily affect layers above and below it. Such architectures are ideal for today’s real-life, healthcare and even business , applications as the changes in business rules and environment do not require changes to the application’s code or to the entire architecture (although some changes in the configuration files may be required).

An n-tier-to-n-tier architecture is proposed, which is suitable for deploying mainstream Grid technologies to semantically integrate heterogeneous Data Sources (DS). By using this architecture, any n-tier application can be easily implemented over the Grid in a distributed manner in order to achieve semantic interoperability among geographically distributed heterogeneous data sources. This architecture can scale out the application’s load and it encompasses the following three main components:

***Component-I:*** Physically Distributed Data Sources (PDDS)

***Component-II:*** Semantic Query Engine (SQE)

***Component-III:*** Web-based User Interface (WUI)

Generally, these components are known to be tiers of a three-tier model. However, in this study all three are referred to as “independent components” (and not “tiers”), in order to avoid any confusion, as each of these subsequently enclose n-

tiers. All the components of this architecture are loosely coupled in a distributed fashion, as shown in Figure 19.

In order to enable the scalability, manageability and agility of applications, it is essential to build them on top of a consistent architecture that serves as a vital foundation. As learnt from the literature review of Grid semantic interoperability (Chapter 3), the novelty of the proposed ASIDS architecture lies in the semantic mapping operation taking place in the Query Processor of Component-II, where a map of the generic and user-defined ontologies is dynamically generated. The distinguished feature of ASIDS is that each of its components and sub-components can be further split into n-tiers to ease application development and enhance usability. It follows a simple yet functional design and is manageable even with increasing number of nodes.

As will be shown later in the evaluation analysis (Chapter 8), the proposed ASIDS architecture provides increased manageability where the number of users is large and reduces workload at the management-level of the system application.



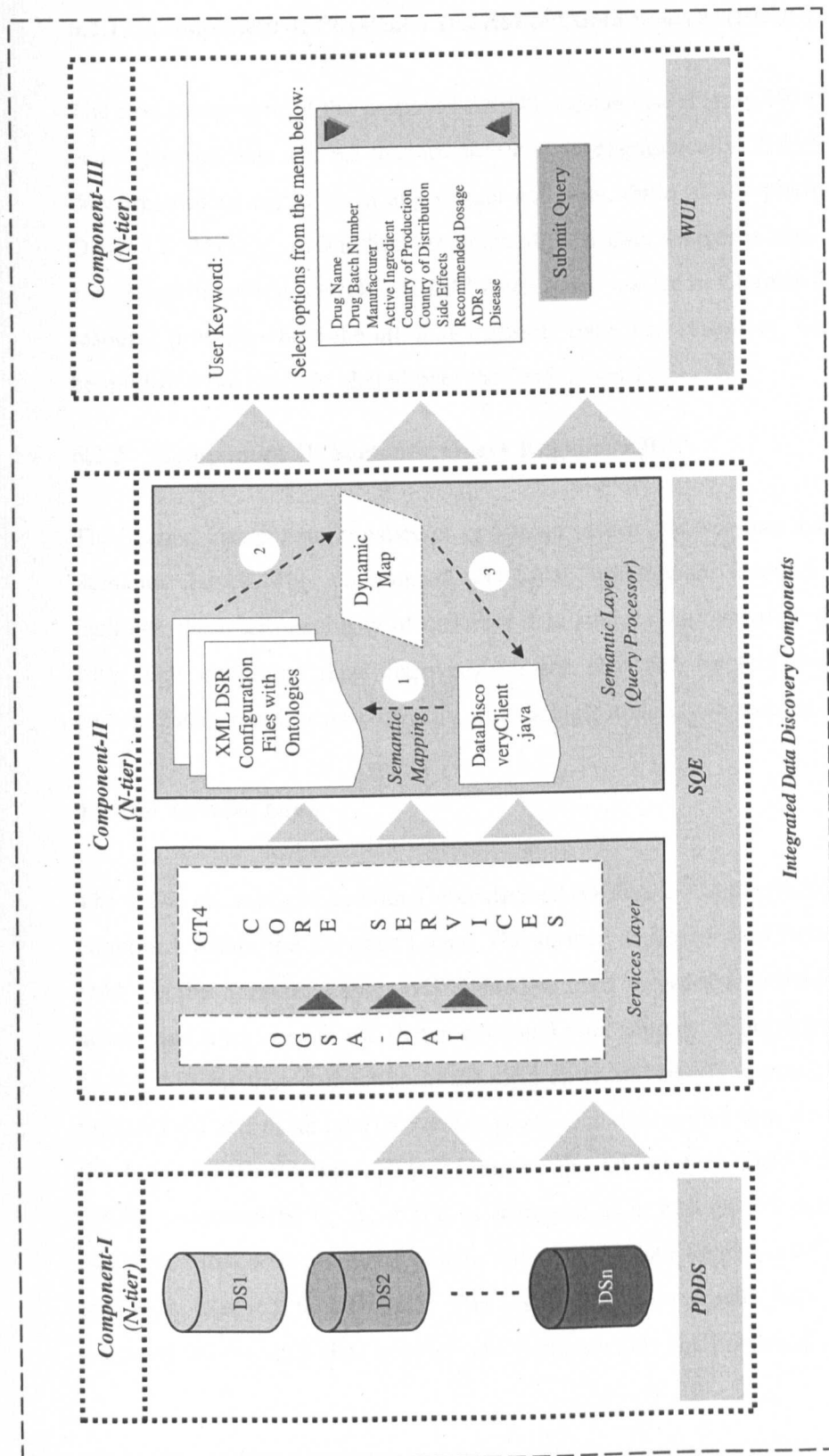


Figure 19: N-Tier-to-N-Tier ASIDS Architecture

### 6.2.1 Component-I: Physically Distributed Data Sources (PDDS)

The first component of the proposed ASIDS architecture (Figure 19) comprises of heterogeneous data sources that are physically (geographically) distributed. These data (re)sources could be in any format and may abide to any platform such as ORACLE, MySQL, or Xindice, etc. Each of these data sources is exposed through a common interface, the OGSA-DAI (mentioned earlier in Chapter 4). The data resource providers hold the ultimate authority over their resources, which can not be modified but are only shared over the Grid network.

### 6.2.2 Component-II: Semantic Query Engine (SQE)

The second component consists of two main layers: the Services Layer and the Semantic Layer. This component acts as a “middleware resource broker” to facilitate the interoperability of different data sources and semantic discovery of data. The integrated data discovery service provided by this middleware is customizable, flexible, user-friendly, quite simple and easy to manage.

#### *a. The Services Layer*

The GT4 core services and other user-defined (OGSA-DAI type) data services are contained within the Services Layer. The service deployment is through OGSA-DAI. At the Services Layer, OGSA-DAI is used in order to provide controlled access and interfaces to the (heterogeneous) data sources; it provides a common JAVA API for their federation in the Grid environment. As shown in Figure 19, OGSA-DAI acts as an interface and exposes each data source that are available on the Grid. In the Globus (GT4) container, the OGSA-DAI Grid Data Services (GDS) (Antonioletti et al., 2005) is deployed as a customized data service to establish a link with the physical data sources. The data sources are first deployed and then exposed to the GDS. The GDS is a user-created data service that accesses the physical data sources and is responsible for providing interfaces to

them. It is through the Grid Data Service (GDS) that various data resources are linked.

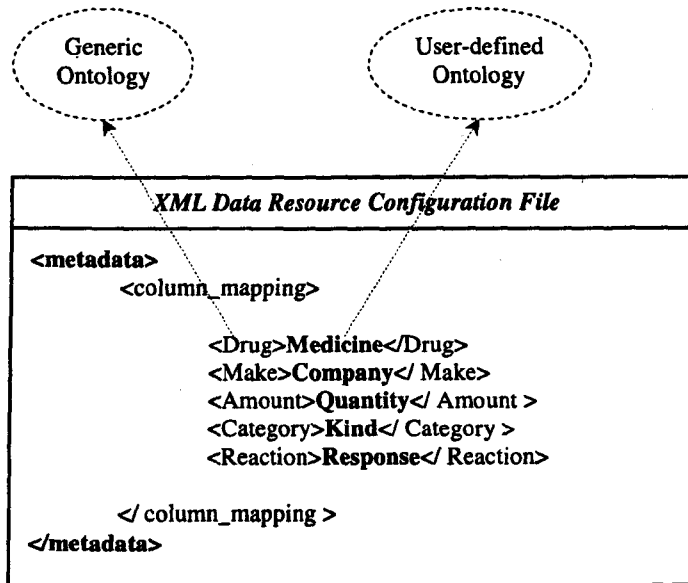
### ***b. The Semantic Layer***

The Query Processor is a sub-component that is contained within the Semantic Layer. At the semantic layer, the framework for Query Processor contains the `DataDiscoveryClient.java`, a Java client Servlet, which is built using the Eclipse 3.2.2 environment and runs on Apache Tomcat Server (version 5.0.28) (Brittain and Darwin, 2003). Once created, the GDS (from the Services Layer) is then accessed by the `DataDiscoveryClient` Servlet in order to fetch data from the distributed data sources. The client Servlet also interacts with the Data Resource Configuration files. These files are XML documents that not only contain information about the metadata of the data sources (DS), such as the product vendor and version, for example the vendor and version information for MySQL 5.0 or Postgres 8.2 would be automatically stored into the DSR configuration files during deployment of the DSRs onto the GDS, the information about other user-defined metadata (attributes of the tables and database schema) is also contained within the DRS configuration files. It is here that users can define generic ontologies and specify them with data resource-specific ontologies, for each field of the database tables (a small chunk from the sample Data Resource Configuration file is shown in Figure 20).

A separate Data Resource Configuration file is created for each data resource, during its deployment upon the OGSA-DAI service (GDS). These files prove to be very useful while retrieving data from multiple tables, which have different (heterogeneous) field-names for entities that are identical in context.

As shown in Figure 20, the metadata tag contains the user-defined metadata specification about a particular data resource. The tag for `<columnMapping>` is a user-defined tag, which indicates a property of the data resource. It further contains the specification matching of generic ontology with the Data Source-

specific (DS-specific) ontology. The DS-specific ontologies differ for each data source as every database table or relation has different names for its columns/attributes (heterogeneous relations have been used), whereas the generic ontology remains constant (common) for each data source.



**Figure 20: Ontology Specification**

In the sample Data Resource Configuration file (as shown in Figure 20), there is a semantic matching of the generic ontology with the DS-specific ontology, such as '<Drug>' with 'Medicine' respectively (which are different "names" but have the same meaning and refer to the same real object/concept – in this case "drug"). Similarly '<Make>' which is a generic ontology term is mapped to 'Company' which is from the DS-specific ontology, '<Amount>' to 'Quantity', etc. can also be dealt within the same context. Here the DS-specific ontologies, as shown in Figure 6, namely: Medicine, Company, Quantity, Kind and Response, are the actual column names (field names) of the relational database tables. By doing such a mapping, the problem of heterogeneous field-names could be addressed and a single SQL data retrieval query could be composed for fetching data from heterogeneous data sources. .

When the client Servlet passes an SQL user query through the GDS to the data resource, the query first goes to the Data Resource Configuration files to dynamically create a semantic map of the data resource metadata ontologies (both generic and DS-specific) and fetches integrated data from all distributed sources, regardless of their semantic heterogeneity. This dynamically created semantic map gets a list of all values that match to the generic-ontology terms. During query processing, when the semantic map is generated dynamically, the ontology specification (Figure 20) illustrates that generic term is “equal to” or has the same meaning as the DS-specific term (e.g. “<Amount>” = “Quantity”), which makes it easier for the query to access and integrate data from all (heterogeneous) fields with the same contextual meaning or semantics.

This operation is explicitly demonstrated in a practical way in the experimental prototype implementation in Chapter 7.

### 6.2.3 Component-III: Web-based User Interface (WUI)

The third component of this architecture is an interface layer for user interaction. It could be built by using any of the web services technologies. The one proposed in the architecture constitutes of a JAVA Server Page (JSP at the front-end) that calls a JAVA Servlet, the DataDiscoveryClient Servlet, upon submitting the query request. This Servlet runs on Apache Tomcat Server (version 5.0.28), which is a Servlet or service container and is an open-source, reliable application solution (Brittain and Darwin, 2003). The results of the user query are fetched and then displayed on the webpage.

## 6.3 Conclusion

An n-tier-to-n-tier application architecture (ASIDS), designed to semantically integrating heterogeneous data resources, has been proposed that can enable feasible implementation to validate the research hypothesis. All the components of this architecture are explained in detail. Each of the Components I-III can be further divided into n-tiers. In this architecture, these components act as layers

that are loosely integrated in a distributed manner. Moreover, ASIDS employs Grid technologies tightly integrated with Web Services technologies, as suggested by Naseer & Stergioulas (Naseer and Stergioulas, 2007).

The ASIDS architecture aims at providing a viable solution to the semantic heterogeneity problems on Grids by resolving semantic heterogeneity among the *data fields* using OGSA-DAI, thus enabling the semantic federation of heterogeneous data resources at the *data field-level* or *attribute-level*. The SQE component of the ASIDS architecture facilitates this semantic integration. For the sake of simplicity, ASIDS does not use any of the industry-developed ontology mapping tools such as PROMPT (Noy and Musen, 2000), ONION (Mitra et al., 2000), Chimaera (McGuinness et al., 2000), FCA-Merge (Stumme and Madche, 2001), GLUE (Doan et al., 2002) and OBSERVER (Mena et al., 2000), etc. and uses *XML Resource Configuration Files* from the OGSA-DAI for ontology mapping which is the novelty of ASIDS. Due to the complex nature of the problem, the ontologies are defined only at the upper-level as generic or reference ontologies. The user-defined or local ontologies are mapped to the generic ontologies in accordance to the underlying semantics of their terminologies.

The ASIDS architecture provides basis for the implementation of the experimental prototype in the next chapter.

## Chapter 7

### Implementation of the Proposed Architecture: Building a Prototype

#### 7.1 Introduction

The next step towards verifying the research hypothesis was to build an experimental prototype, implementing the proposed ASIDS architecture (proposed in Chapter 6), and to portray its functionality.

This chapter presents the implementation of the ASIDS, in the form of a HealthGrid application (experimental prototype), which will be henceforth called the ASIDSApplication, which was used to verify the hypothesis. This chapter first sets up the ASIDSApplication context, then describes the various stages of prototype development, and finally discusses the features and functionalities of the developed application.

#### 7.2 Application Context

It is important to have in mind the kind of semantic matching used and the level of data integrity considered in this research. In general, the level of data integrity is based on the level of heterogeneity of the various data resources used and includes syntactic heterogeneity (i.e. differences in the data models and data types, which can be easily resolved) and semantic heterogeneity (i.e. differences in the underlying meaning of data, which play an essential role during heterogeneous resource integration and interoperation) (Verschelde and Dos, 2004). The work of this thesis aims at resolving the *semantic* heterogeneity, which mainly deals with the meanings of the data fields.

Thus, the contribution of this thesis is an approach that uses contemporary Grid technologies for integrating heterogeneous data resources that have semantically different data fields (attributes). The approach is demonstrated using a prototype HealthGrid. Without significant extra effort, this approach can be applied to address syntactic heterogeneity.

This heterogeneity of the data resources makes semantic integration an extremely complex problem. Ontology matching is an important component of semantic integration and the mismatch between ontologies is found firstly at the language-level (for different languages that have different semantics) and secondly at the ontology-level (within the same language), e.g. using same linguistic term to describe different concepts or using different term to describe the same concept, etc. in order to facilitate semantic integration. The various tools for automatic and semi-automatic ontology mapping use concept names and natural language descriptions as a feature in their ontology definitions (Noy, 2004).

As seen from the literature survey in Chapter 3 and Chapter 4 and from the context of the problem in the previous chapters, data integration is one of the major challenges faced today in many sectors. This problem becomes magnified if the data resources are heterogeneous in some way or at some level such as platform-level, format-level or field-level, etc. The ASIDS architecture aims at providing a viable solution to such problems by enabling semantic federation of data resources.

The application domain chosen for this pilot study is healthcare, in particular the Pharmaceutical sector, where data integrity and platform compatibility are critical to the provision of consistent medical information to the various healthcare stakeholders.

Therefore, ASIDSApplication - a HealthGrid-enabled application prototype, was developed, based on the proposed ASIDS architecture, which collects Pharmaceutical (drug) data from various data resources regardless of their



geographical locations, heterogeneous formats and different (field-level) semantics, and makes this data available to the users (such a HealthGrid is called a PharmaGrid, and can be very useful in integrating heterogeneous pharma information, for example in Pharmacovigilance). The data retrieved is in XML format, which can be further used or converted into any other format accordingly, to perform desired operations such as simulation and modelling of information or comparative analysis of results in order to carry out collaborative research by sharing information contained in heterogeneous data sources.

### **7.3 Prototype Development**

For the prototype development, the Rapid Prototyping Model was used as the software engineering process model, as it is considered to be an appropriate model. The prototyping was done in three sub-phases namely: build, modify, and test (the last two being in an iterative loop), to attain correctness or consistency in the prototype design and functionality. The built, modify and test loop was repeated few times until the prototype was operating successfully.

#### **7.3.1 Setting up the Prototype Application Environment**

Before building the prototype it was necessary to set up the ASIDSApplication environment; therefore necessary installations of various Grid technologies such as GT4 and OGSA-DAI were made. However, it was first needed to download them (see Appendix-A). The setting up of the prototype environment consisted of four steps that are described further.

##### ***a. Setting up the Globus Container (GT4 & OGSA-DAI)***

Once the necessary installations were completed, and the system configuration settings were finalised, the available Grid services were checked by running the commands in the terminal window (see Figure 21). To see the list of installed core Grid services, the command '*Globus-start-container*' was run from the terminal. As shown in Figure 21, a list of available Grid services was displayed. The last

service number 25 (highlighted) is an OGSA-DAI data service that is running in the GT4 container. It is a user-defined service that was deployed on the GT4 container and was named as DataDiscovery1. This is the GDS (Grid Data Service) handle that acts as the interface and is used to access the distributed data sources in the HealthGrid prototype.

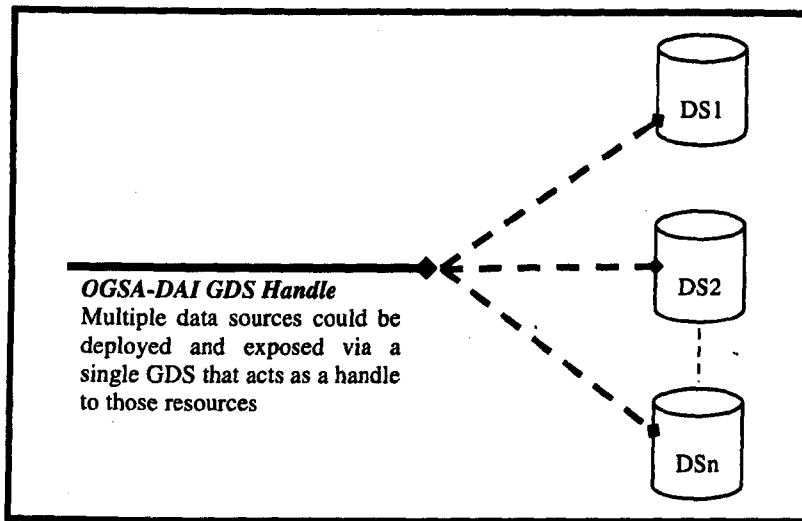
```

root@laptop:/usr/local/globus
File Edit View Terminal Tabs Help
[root@laptop globus]# globus-start-container -nosec
Starting SOAP server at: http://127.0.0.1:8080/wsrf/services/
With the following services:

[1]: http://127.0.0.1:8080/wsrf/services/AdminService
[2]: http://127.0.0.1:8080/wsrf/services/AuthzCalloutTestService
[3]: http://127.0.0.1:8080/wsrf/services/ContainerRegistryEntryService
[4]: http://127.0.0.1:8080/wsrf/services/ContainerRegistryService
[5]: http://127.0.0.1:8080/wsrf/services/CounterService
[6]: http://127.0.0.1:8080/wsrf/services/ManagementService
[7]: http://127.0.0.1:8080/wsrf/services/NotificationConsumerFactoryService
[8]: http://127.0.0.1:8080/wsrf/services/NotificationConsumerService
[9]: http://127.0.0.1:8080/wsrf/services/NotificationTestService
[10]: http://127.0.0.1:8080/wsrf/services/PersistenceTestSubscriptionManager
[11]: http://127.0.0.1:8080/wsrf/services/SampleAuthzService
[12]: http://127.0.0.1:8080/wsrf/services/SecureCounterService
[13]: http://127.0.0.1:8080/wsrf/services/SecurityTestService
[14]: http://127.0.0.1:8080/wsrf/services/ShutdownService
[15]: http://127.0.0.1:8080/wsrf/services/SubscriptionManagerService
[16]: http://127.0.0.1:8080/wsrf/services/TestAuthzService
[17]: http://127.0.0.1:8080/wsrf/services/TestRPCService
[18]: http://127.0.0.1:8080/wsrf/services/TestService
[19]: http://127.0.0.1:8080/wsrf/services/TestServiceRequest
[20]: http://127.0.0.1:8080/wsrf/services/TestServiceWrongWSDL
[21]: http://127.0.0.1:8080/wsrf/services/Version
[22]: http://127.0.0.1:8080/wsrf/services/WidgetNotificationService
[23]: http://127.0.0.1:8080/wsrf/services/WidgetService
[24]: http://127.0.0.1:8080/wsrf/services/gsi/AuthenticationService
[25]: http://127.0.0.1:8080/wsrf/services/ogsadai/DataDiscovery1
  
```

Figure 21: GT4 Services Container

OGSA-DAI (Open Grid Services Architecture Data Access and Integration) was used as an interface for first deploying and then exposing the heterogeneous data sources to the Grid Data Services (GDS), as shown in Figure 22.



**Figure 22: Interface to Data Sources**

The GDS is a user-created data service that accesses the physical data sources and is responsible for providing interfaces to them.

***b. Setting up the Data Sources (Testbed for a Health-DataGrid)***

After completing the installations and necessary settings, the experimental databases (data sources) were constructed, taking into consideration the standard fields that are used in professional medical and/or Pharmaceutical database systems. These databases comprise the testbed for this study. Six (experimental) heterogeneous data sources were built using MySQL, three on each of the LINUX machines. One reason for using MySQL was that it is open-source, easy and simple to setup; another reason was that many existing medical databases are built using MySQL. Moreover, it would be easier to setup this prototype as a healthcare or business application in the real-world domain. For the sake of simplicity, only one relation (table) per data source was created. Each relation is uniquely identified by a unique identifier (UID-Primary Key). This UID plays no operational role in the functionality of the prototype, but is used for checking the consistency of the data sources created. However, it could be used for extended functionalities of the application, if needed, in future research.

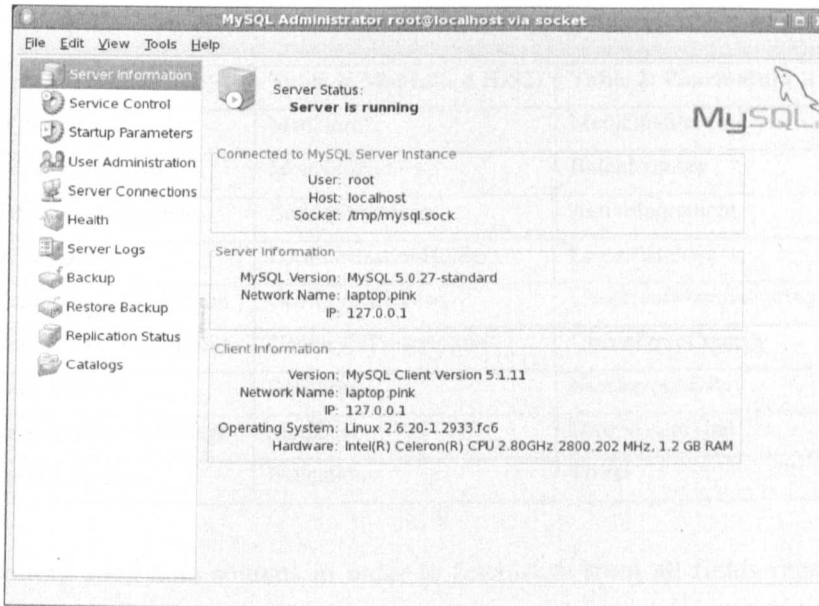


Figure 23: GUI for MySQL Administrator

MySQL GUI Tools were used for creating and managing the data sources (databases and tables). These are open-source and can be downloaded from the MySQL website (Appendix-A). For this HealthGrid experimental prototype, MySQL Query Administrator (MQA) (Figure 23) was used in order to create all six data sources. MQA is a user-friendly GUI-based tool for managing databases such as for creating, deleting and modifying tables.

Heterogeneity of the six data sources was ensured in terms of their heterogeneous fields. The field labels of all the data sources were different, but they contained the same (type of) information as shown in Table 3. For instance, the field labelled as 'D\_Name' of the first data source (DS1) contains the same type of information (i.e. represents the same real entity) as the field labelled 'MedNom' of the second data source (DS2) and the field labelled 'MedicinalProduct' of the third data source (DS3). All three of these fields (columns) represent the same type of information (i.e. name of the medicine or drug) but under different labels.

Table 3: Fields of Three Data Sources Having Different Names but Containing the Same Type of Information

Table 1: DInfo_1 (DS1)	Table 2: MedInfo_1 (DS2)	Table 2: PharmaInfo_1 (DS3)
D_Name	MedNom	MedicinalProduct
License_Number	MedNumber	BatchNumber
Effective_Ingredient	ActiveSubstance	ActiveIngredient
Manufacturer	DAuthorizationHolder	LicenseHolder
Country_of_Production	CountryofLicense	CountryofManufacturing
Country_of_Distribution	CountryofPrescription	ObtainDrugCountry
Side_Effects	Reactions	NumberofADRs
Recommended_Dosage	MedDose	DrugDosageUnit
Target_Disease	Indications	Treats

Federating such data sources in order to fetch data from all fields regardless of their different ontologies was not a straightforward task. However, this was made feasible by using the ASIDS architecture. The Figure 24, Figure 25 and Figure 26 show the GUI of the MQA table editor, showing fields for the tables “DInfo\_1” (DS1), “MedInfo\_1” (DS2) and “PharmaInfo\_1” (DS3), respectively.

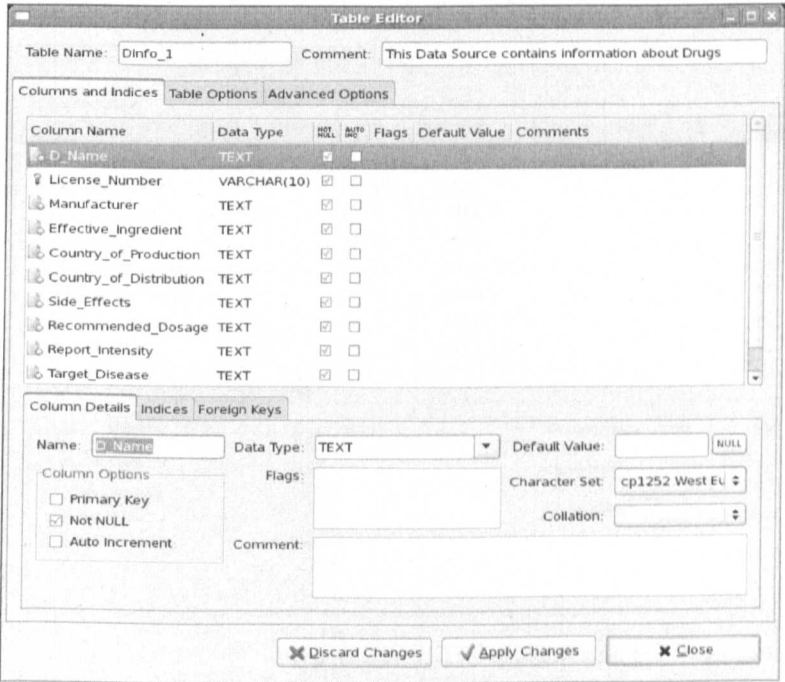


Figure 24: MQA GUI of Table Editor Showing Fields for the Table “DInfo\_1” (DS1)

Table Editor

Table Name: MedInfo\_1 Comment: This Data Source contains information about Medicines

Columns and Indices Table Options Advanced Options

Column Name	Data Type	NOT NULL	Flags	Default Value	Comments
MedNom	TEXT	<input type="checkbox"/>	<input type="checkbox"/>	NULL	
MedNumber	VARCHAR(25)	<input checked="" type="checkbox"/>	<input type="checkbox"/>		
ActiveSubstance	TEXT	<input type="checkbox"/>	<input type="checkbox"/>	NULL	
DAuthorizationHolder	TEXT	<input type="checkbox"/>	<input type="checkbox"/>	NULL	
Reactions	TEXT	<input type="checkbox"/>	<input type="checkbox"/>	NULL	
CountryofLicense	TEXT	<input type="checkbox"/>	<input type="checkbox"/>	NULL	
CountryofPrescription	TEXT	<input type="checkbox"/>	<input type="checkbox"/>	NULL	
MedDose	TEXT	<input type="checkbox"/>	<input type="checkbox"/>	NULL	
Indications	TEXT	<input type="checkbox"/>	<input type="checkbox"/>	NULL	

Column Details Indices Foreign Keys

Name: MedNom Data Type: TEXT Default Value: NULL NULL

Column Options: ☐ Primary Key ☐ Not NULL ☐ Auto Increment

Flags: Character Set: cp1252 West Eu Collation:

Comment:

Discard Changes Apply Changes Close

Figure 25: MQA GUI of Table Editor Showing Fields for the Table “MedInfo\_1” (DS2)

Table Editor

Table Name: PharmaInfo\_1 Comment: Data Source contains information about Pharma Products

Columns and Indices Table Options Advanced Options

Column Name	Data Type	NOT NULL	Flags	Default Value	Comments
MedicinalProduct	TEXT	<input type="checkbox"/>	<input type="checkbox"/>	NULL	
BatchNumber	VARCHAR(25)	<input checked="" type="checkbox"/>	<input type="checkbox"/>		
ActiveIngredient	TEXT	<input type="checkbox"/>	<input type="checkbox"/>	NULL	
LicenseHolder	TEXT	<input type="checkbox"/>	<input type="checkbox"/>	NULL	
NumberOfADRs	TEXT	<input type="checkbox"/>	<input type="checkbox"/>	NULL	
CountryofManufacturing	TEXT	<input type="checkbox"/>	<input type="checkbox"/>	NULL	
ObtainDrugCountry	TEXT	<input type="checkbox"/>	<input type="checkbox"/>	NULL	
DrugDosageUnit	TEXT	<input type="checkbox"/>	<input type="checkbox"/>	NULL	
Treats	TEXT	<input type="checkbox"/>	<input type="checkbox"/>	NULL	

Column Details Indices Foreign Keys

Name: MedicinalProduct Data Type: TEXT Default Value: NULL NULL

Column Options: ☐ Primary Key ☐ Not NULL ☐ Auto Increment

Flags: Character Set: cp1252 West Eu Collation:

Comment:

Discard Changes Apply Changes Close

Figure 26: MQA GUI of Table Editor Showing Fields for the Table “PharmaInfo\_1” (DS3)

The MySQL Query Browser (MQB) was used to populate the data sources with different values. MQB is a user-friendly GUI-based tool for inserting, modifying and deleting records (rows) from the database table.

Figure 27, Figure 28 and Figure 29 show three data sources, namely DS1, DS2 and DS3 respectively, with data populated in the tables. Hence, the data sources were first created using the MQA GUI interface and then rows were inserted in them using the MQB GUI interface.

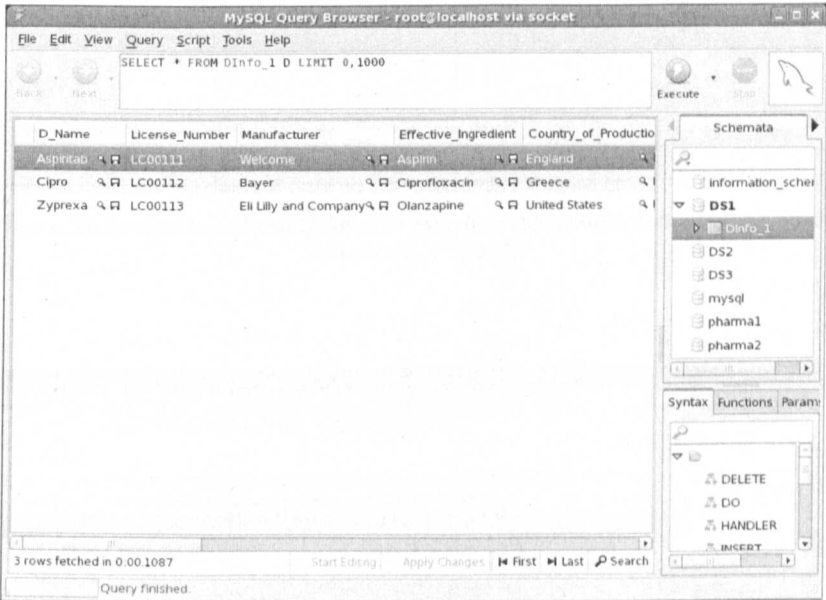
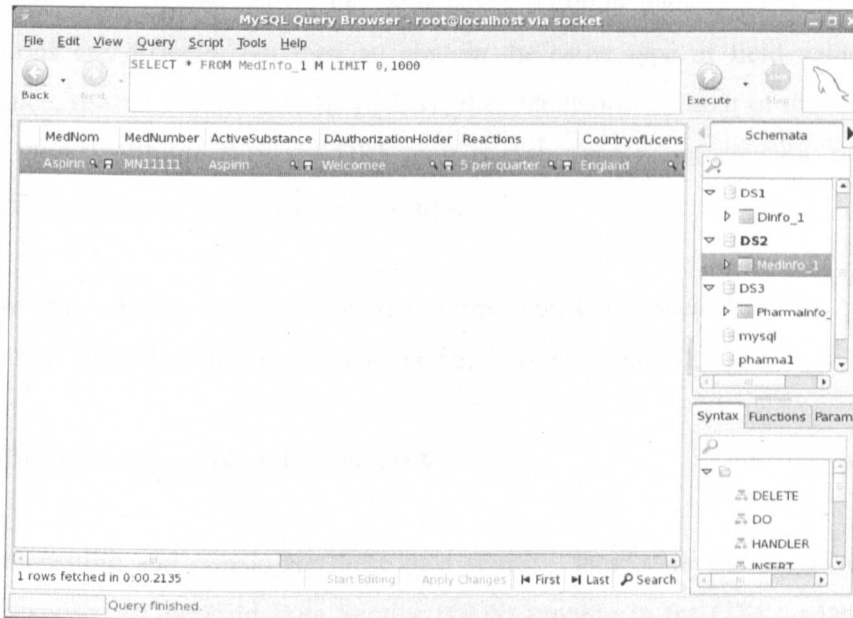
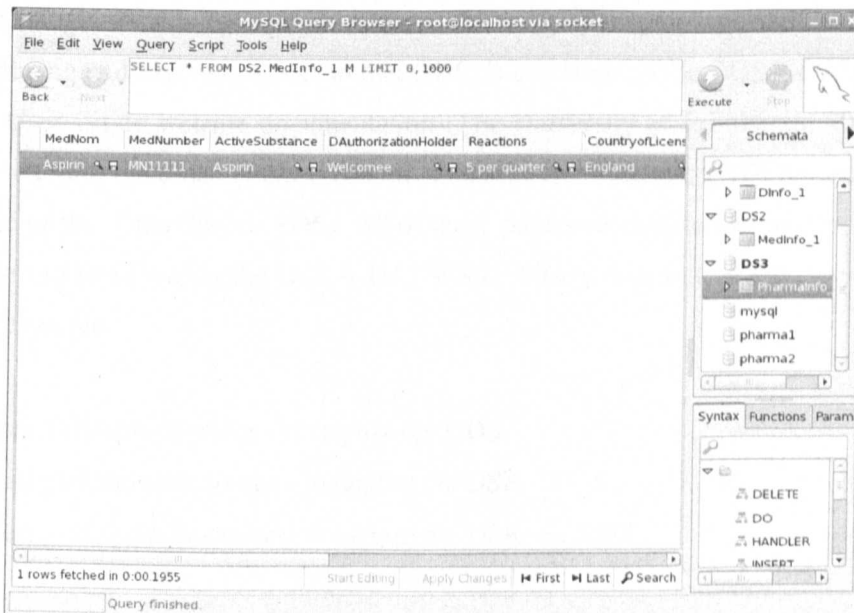


Figure 27: Screenshot of the MQB GUI interface that Shows Values in the Table "DInfo\_1" (DS1)



**Figure 28: Screenshot of the MQB GUI interface that Shows Values in the Table "MedInfo\_1" (DS2)**



**Figure 29: Screenshot of the MQB GUI interface that Shows Values in the Table "PharmaInfo\_1" (DS3)**



As seen from figures above, the field labels (column names) for all the three relations are different but they all contain the same type of field values. For instance, the 'D\_Name' column of DS1, the 'MedNom' column of the DS2 and the 'MedicinalProduct' of the DS3 contain same type of information, which is about the names of the medicines or drugs.

These data sources make up the first component, Component-I (PDDS), of the proposed ASIDS architecture (mentioned earlier in Chapter 6).


### *c. Configuration of the GDS and DSR*

After creating and populating these data sources, they were first deployed and then exposed to the Grid Data Service (GDS) running in the GT4 container, as shown in Figure 21. As mentioned earlier, GDS is a user-defined data service that accesses the physical Data Service Resources (DSRs) and is responsible for providing interfaces to them. There could be multiple DSRs deployed or exposed to a single GDS. The entire process for initially deploying GDS, and then deploying and exposing DSRs to that GDS, is available on the OGSA-DAI WSRF 2.2 User Guide website documentation (The University of Edinburgh, 2006). In this website documentation, the term 'Data Service Resource' (DSR) is used in place of the 'Data Source' (DS). All of these processes require the 'ant' command to be run from within the OGSA-DAI WSRF binary distribution directory (Bin). For example:

- ant guiDeployService - to deploy the GDS
- ant guiDeployResource - to deploy the DSR
- ant guiExposeResource - to expose the DSR via GDS

Once the GDS is deployed and the DSRs are deployed and exposed successfully, a list of all DSRs that are exposed via a specific GDS can be seen with the following command (Figure 30):

ant listResourcesClient -Ddai.url = SERVICE-URL (complete URL of the GDS as shown in Figure 21).



```

root@laptop:/usr/local/ogsadai
File Edit View Terminal Tabs Help
[root@laptop ~]# cd /usr/local/ogsadai
[root@laptop ogsadai]# ant listResourcesClient -Ddai.url=http://localhost:8080/wsrf/services/ogsadai/DataDiscovery1
Buildfile: build.xml

setupClientSecurity:

listResourcesClient:
  [java] Service version: OGSA-DAI WSRF 2.2
  [java] Number of resources: 4
  [java] Resource: DSR_DS2
  [java] Resource: DISC_DSR_DS1
  [java] Resource: DSR_DS1
  [java] Resource: DSR_DS3

BUILD SUCCESSFUL
Total time: 10 seconds
[root@laptop ogsadai]#

```

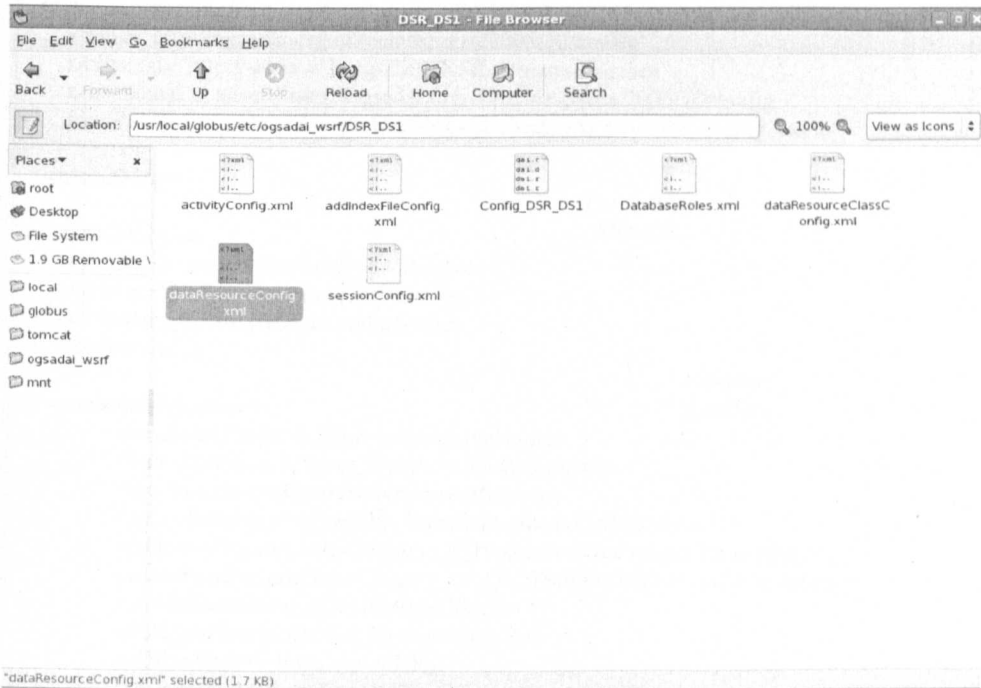
**Figure 30: List of DSRs Exposed via DataDiscovery1**

Figure 30 shows a list of all the DSRs exposed via the user-defined GDS named “DataDiscovery1”.

#### *d. Setting up the Resource Configuration Files*

To setup the mapping between generic ontologies and DS-specific ontologies, the Data Resource Configuration file (dataResourceConfig.xml) file needs to be altered for each of the DSRs created. The Data Resource Configuration file is an XML document that contains complete information about the DSR, such as its name, vendor, version, attributes, etc. It is here that the custom metadata or semantics are defined and generic ontologies are matched against the DSR-specific ontologies. Inside the Globus directory, there is a separate directory for each DSR, where the dataResourceConfig.xml file of the respective resource resides (see Figure 31), so for every DSR (DS1, DS2, DS3, etc.) there is one individual dataResourceConfig.xml file.

Figure 31 shows dataResourceConfig.xml file, which is contained inside the DS1 folder/directory, for one of the DSRs (DS1). For the purpose of ontology mapping, this file needs to be modified for each of the DSRs individually.



**Figure 31: Location of the dataResourceConfig.xml File for DS1**

As shown in Figure 32, the metadata tag contains the pre-defined metadata and custom or user-defined metadata specification about a particular data resource. The tag for `<productInfo>` is a pre-defined tag, which is there by default in each of the Data Resource Configuration files for every DSR. Both the tags for `<columnMapping>` and `<tableMapping>` are custom metadata or user-defined tags, which indicate properties of the data resource. The `<columnMapping>` tag further contains the specification matching between generic ontology and the Data Source-specific (DS-specific) ontology (see Figure 33). The DS-specific ontologies differ for each data source, since every database table/relation has different names/labels for its columns/attributes, whereas the generic ontology (in bold) remains constant for each data source (also see Figure 34 and Figure 35).

```

<?xml version="1.0" encoding="UTF-8"?>

<!-- (c) International Business Machines Corporation, 2002 - 2005. -->
<!-- (c) University of Edinburgh 2002 - 2005. -->
<!-- See OGSA-DAI-Licence.txt for licensing information. -->

<dataResourceConfig
  xmlns="http://ogsadai.org.uk/namespaces/2005/10/config"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://ogsadai.org.uk/namespaces/2005/10/config
file:///usr/local/globus/share/schema/ogsadai/xsd/data_resource_config.xsd">

  <metaData>
    <productInfo>
      <productName>MySQL</productName>
      <productVersion>4.0</productVersion>
      <vendorName>MySQL</vendorName>
    </productInfo>
    <columnMapping>
      <medicineName>D_Name</medicineName>
      <batchNumber>License_Number</batchNumber>
      <manufacurer>Manufacturer</manufacurer>
      <activeSubstance>Effective_Ingredient</activeSubstance>
      <countryofProduction>Country_of_Production</countryofProduction>
      <countryofPrescription>Country_of_Distribution</countryofPrescription>
      <sideEffects>Side_Effects</sideEffects>
      <dosage>Recommended_Dosage</dosage>
      <aDRs>Report_Intensity</aDRs>
      <recommendedFor>Target_Disease</recommendedFor>
    </columnMapping>
    <tableMapping>
      <tableName>DInfo_1</tableName>
    </tableMapping>
  </metaData>

  <roleMap name="Name"
    implementation="uk.org.ogsadai.common.rolemap.SimpleFileRoleMapper"
    configuration="/usr/local/globus/etc/ogsadai_wsrf/DSR_DS1/DatabaseRoles.xml"/>

  <dataResource>
    <driver implementation="org.gjt.mm.mysql.Driver">
      <uri>jdbc:mysql://localhost:3306/DS1</uri>
    </driver>
  </dataResource>
</dataResourceConfig>

```

Figure 32: DSR Configuration File (XML) for DS1

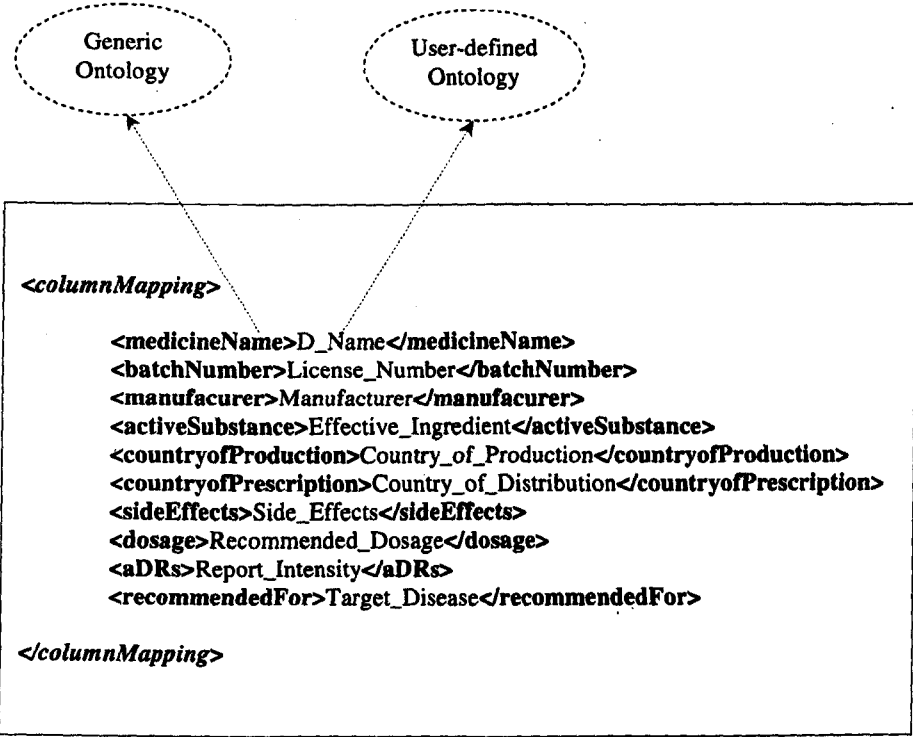


Figure 33: Semantic Ontology Matching for DS1

There is a semantic matching of the generic ontology with the DS-specific ontology for each attribute of the table such as for DS1 (Figure 33), the attribute `<medicineName>` is mapped with 'D\_Name', the attribute `<batchNumber>` is mapped with 'Licence\_Number' and so on (which are different labels but can be used within the same context).

Similarly for DS2 (Figure 34) and DS3 (Figure 35), the generic ontologies were matched with the DS-specific ontologies. It is worth noting that the generic ontology remains the same for every data resource. Obviously, by doing so, the problem of federating semantically heterogeneous data resources can be tackled.

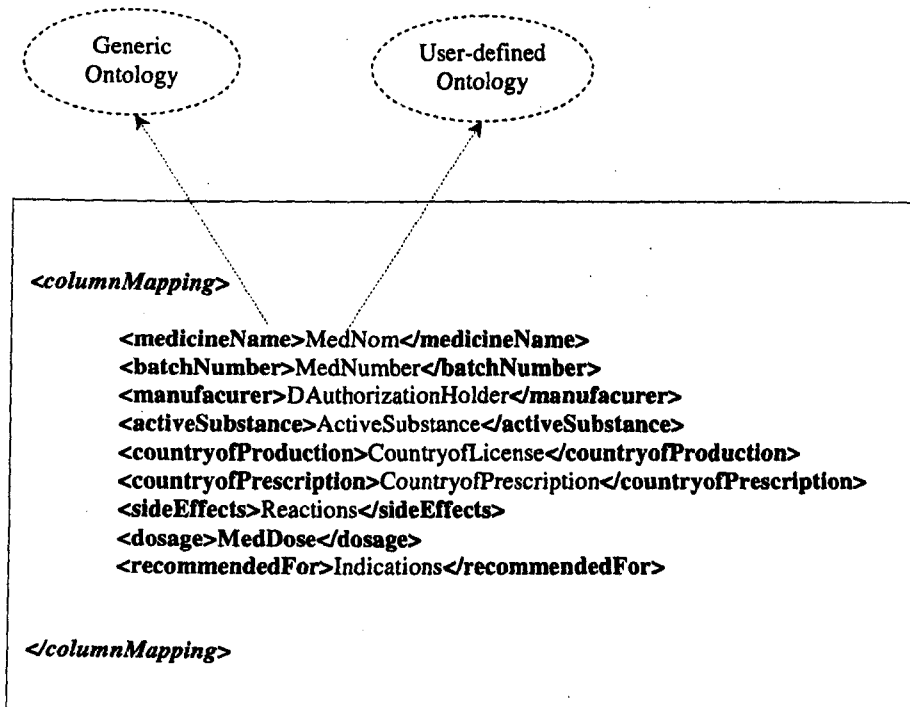


Figure 34: Semantic Ontology Matching for DS2

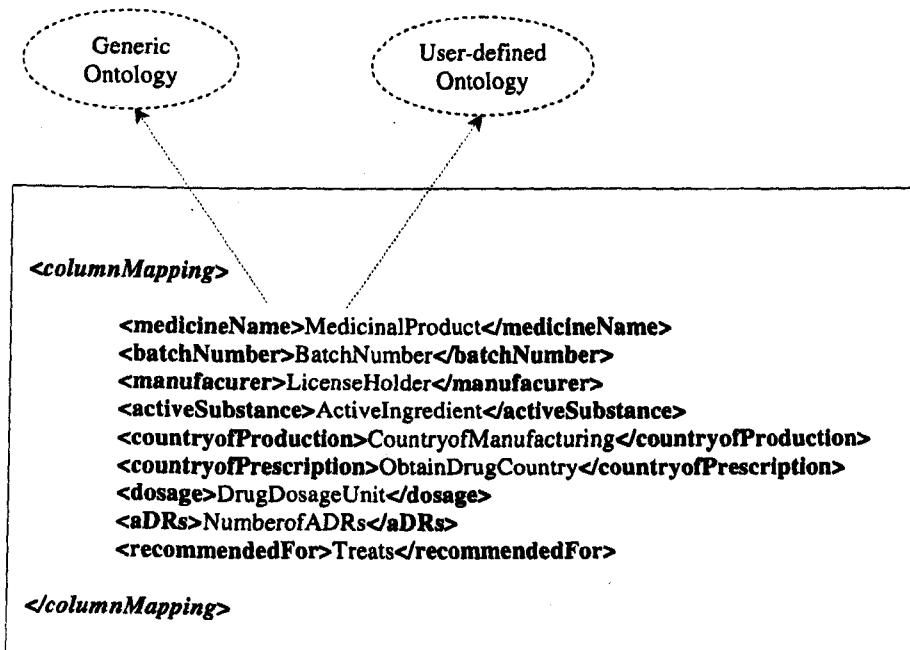


Figure 35: Semantic Ontology Matching for DS3

### 7.3.2 Building the Prototype

After setting-up the ASIDSApplication environment (as described above), a web-based (HealthGrid prototype) project named 'ASIDSApplication' was created using JAVA JDK1.5.0\_09, software development kit (Eclipse SDK 3.2.2). This ASIDSApplication consists of three main components: a JSP page (called as DDQuery.jsp), a client Servlet (called as DataDiscoveryClient.java) and a Java class for semantic mapping (called as Mapping.java). Each of these components is further discussed here.

#### a. JSP Component (DDQuery.jsp)

A simple user interface, JSP page, was produced (Figure 36), which constituted Component-III (WUI) of the proposed ASIDS architecture as mentioned previously in Chapter 6 (Figure 19). The JSP page is used for getting the user's queries. The user query is composed of a (key, value) pair. The user enters a keyword and then selects an item from the menu list. Upon submitting, the query is sent to the Servlet component (DataDiscoveryClient Servlet) of the ASIDSApplication.

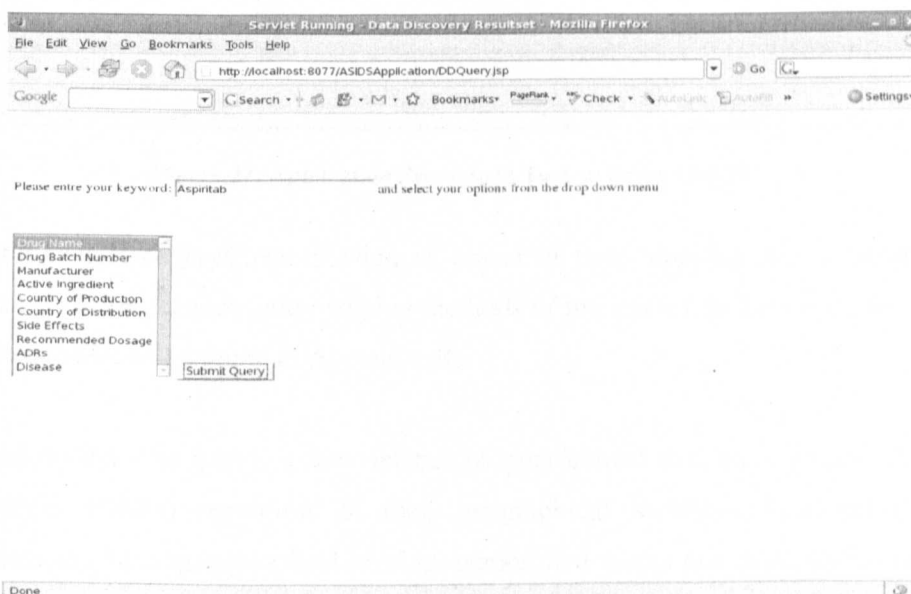
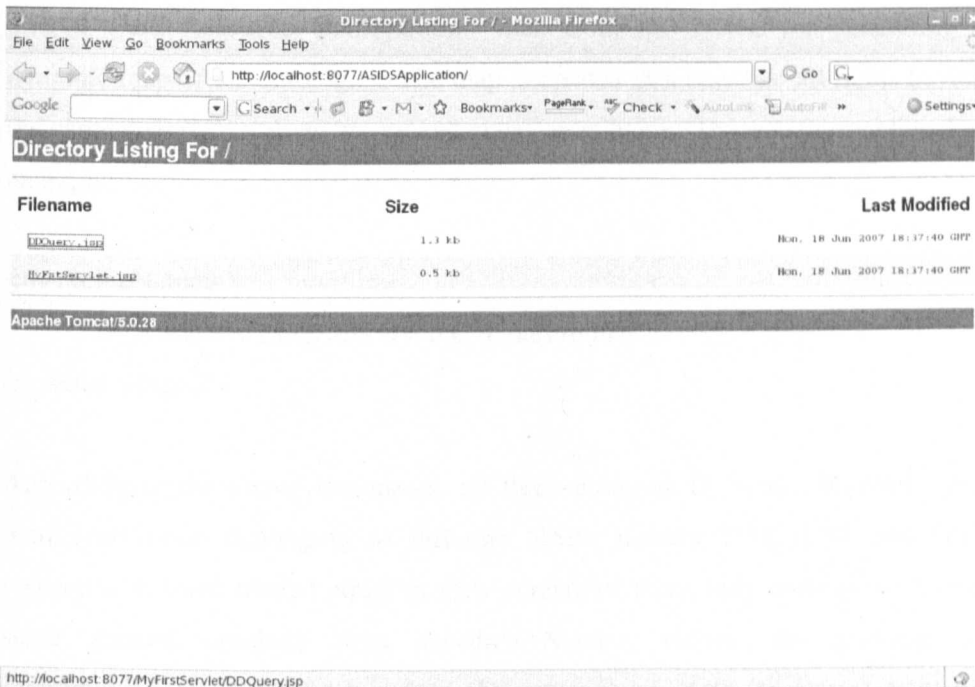


Figure 36: GUI User Interface (JSP Page)

### b. Servlet Component (*DataDiscoveryClient.java*)

When the query goes to the *DataDiscoveryClient.java* Servlet, it in turn accesses the GDS handle, and the semantic matchmaking methods from the *Mapping.java* class are invoked. The *DataDiscoveryClient.java* Servlet needs a Servlet container to run on. Figure 37 shows that the *ASIDSApplication* is running on the Apache Tomcat Server (version 5.0.28) at port 8077.



**Figure 37: Application Running in Tomcat Server 5.0.28**

In the *DataDiscoveryClient* Servlet, an object of type *Mapping.java* is created, which calls the column/field-mapping methods of this class (the Java code for all these classes can be found in Appendix-B).

Based on the user query, it then fetches pharmaceutical data from various data resources (DSRs) regardless of their geographical locations, heterogeneous formats and heterogeneous field-level semantics, and makes this data available on



the HealthGrids. The Servlet then prints the results on the webpage after converting them from XML format into the HTML format.

### c. *Semantic Mapping Component (Mapping.java)*

It is here, in the Mapping.java class, that a dynamic (semantic) map of the generic and DS-specific ontologies is created at run time. When the semantic matchmaking methods from the Mapping.java class are invoked, they access the altered Data Resource Configuration files from all DSRs, for generating a dynamic map of the ontologies. The map is created in a way that all fields having the same generic ontology are ranked equal or as having the same semantics. For example:

```
<medicineName>
    D_Name == MedNom == medicinalProduct
</medicineName>
```

According to the above statements, all three columns D\_Name, MedNom and medicinalProduct belonging to different DSRs namely DS1, DS2 and DS3 respectively, were treated equal in their semantics since they correspond to the same generic ontology term <medicineName>. Hence, the problem of *semantically federating networked (heterogeneous) data resources* can be resolved in Grid environments. Figure 39 shows the results fetched.

The data retrieval results were generally given in the XML format. This format can be further used or converted into any other formats accordingly, as shown in Figure 39; the prototype results have been converted into HTML format for convenience. For this purpose, the XML Style Sheet transform file (Figure 38) was used.

```

<?xml version="1.0"?>
<!-- This is a XSL-Transform that can be used in conjunction with the
xslTransform_from_rowset.xml perform document to transform the results of an SQL
query on the relational database tables into HTML -->
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
xmlns:wrs="http://java.sun.com/xml/ns/jdbc" version="1.0">
<xsl:output method="html" indent="yes"/>
<xsl:template match="/">

    <h2>Query Results</h2>
    <h3>Your query generated the results as below:</h3>

    <table border="1">

        <tr bgcolor="#99CCFF">

            <xsl:for-each select="wrs:webRowSet/wrs:metadata/wrs:column-
definition/wrs:column-label">
                <th><xsl:value-of select="."/></th>
            </xsl:for-each>

        </tr>
        <xsl:for-each select="wrs:webRowSet/wrs:data/wrs:currentRow">

            <tr>
                <xsl:for-each select="wrs:columnValue">
                    <td><xsl:value-of select="."/></td>
                </xsl:for-each>
            </tr>

        </xsl:for-each>

    </table>

</xsl:template>
</xsl:stylesheet>

```

**Figure 38: XML Style Sheet Transform File used to transform the Query Results from XML Format into HTML Format**

The screenshot shows a web browser window titled 'Query Search Results - Windows Internet Explorer'. The address bar shows 'C:\Documents and Settings\cspgaan\Desktop\tempresult.html'. The page content is titled 'Query Results' and states 'Your query generated the results as below:'. It displays three tables of drug information.

MedNom	MedNumber	ActiveSubstance	DAuthorizationHolder	Reactions	CountryofLicense	CountryofPrescription	MedDose	Indications
Aspirino	MN11001	Aspirin	Welcomee	5 per quarter	England	France	1 tab per 4 hrs	Various types of aches, reduces the risk of heart attacks

Drug_Name	License_Number	Manufacturer	Effective_Ingredient	Country_of_Production	Country_of_Distribution	Side_Effects	Recommended_D
Aspirinab	LC00111	Welcome	Aspirin	England	China	Heartburn, nausea, upset stomach	one tablet every 4 h

MedicinalProduct	BatchNumber	ActiveIngredient	LicenseHolder	NumberofADRs	CountryofManufacturing	ObtainDrugCountry	DrugDosageUnit
Aspirina	BNum000001	Aspirin	Welcomea	5 per year	England	Italy	4 tablets per day

Figure 39: Semantic Query Results

Both the DataDiscoveryClient.java Servlet and Mapping.java class constitute Component-II (SQE) of the proposed ASIDS architecture as mentioned earlier in Chapter 6 (Figure 19).

## 7.4 Operational Flow of the Prototype

This section presents the operational flow of the ASIDSApplication (HealthGrid prototype), which is depicted in Figure 40. There are 9 steps involved in the operational flow, from submitting the query until the data is retrieved. These steps are listed below:

- i. The user enters the search query through the browser (JSP Page)
- ii. Then this user query is sent to the Servlet
- iii. The Servlet then accesses the Mapping.java class and invokes its methods to retrieve a list of the DSRs exposed to the OGSA-DAI GDS service

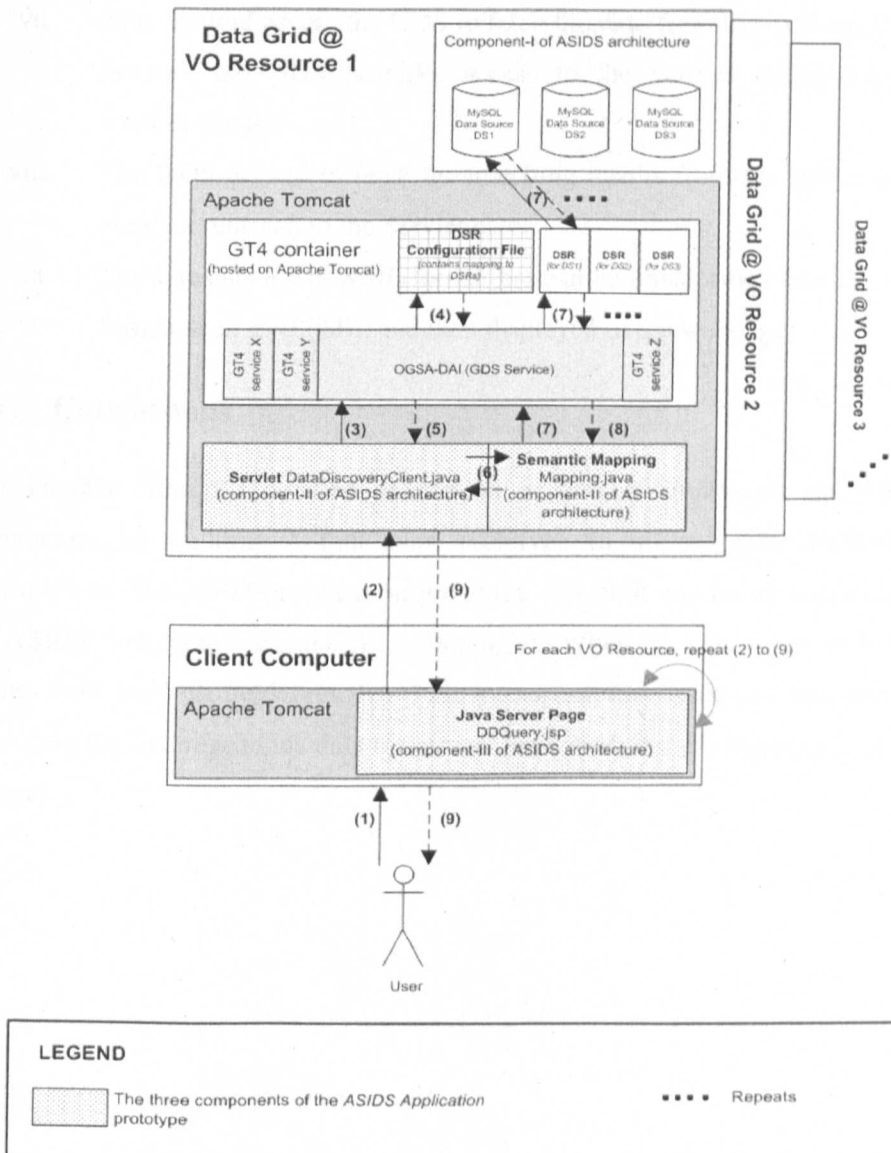


Figure 40: Operational Flow of the Prototype

- iv. The GDS also retrieves a semantic map from the Data Resource Configuration file (dataResourceConfig.xml), that is dynamically generated
- v. This semantic map is then forwarded to the Servlet
- vi. The Servlet then invokes the performQuery method of the Mapping.java class

- vii. This method access the GDS to fetch the data from the exposed DSRs, because the GDS provides access to the various distributed data sources (DSRs)
- viii. The GDS is used to fetch the matching results from the DSRs, which are then returned to the Servlet class
- ix. These results are in XML format but can be transformed into any other format such as HTML, and then displayed to the webpage

## **7.5 Conclusions**

This chapter offered a detailed description of how to implement the ASIDS architecture by building a functional prototype in an exemplar HealthGrid environment. The ASIDSApplication prototype was built and set up according to the ASIDS architecture using Grid technologies. When the user query is sent, it fetches data from all fields, resolving the interoperability issue and semantically federating the heterogeneous data resources, thus verifying the hypothesis of this research.

## Chapter 8

### Evaluation of the ASIDS

#### 8.1 Introduction

An evaluation of the ASIDS architecture is conducted using the implemented experimental prototype. The experimental set-up for this evaluation is described in this chapter and the evaluation results are presented graphically and discussed.

The objective of conducting these experiments was to test the implementation of the ASIDS architecture on the HealthGrid prototype and to demonstrate the feasibility of the proposed approach with *single Grid Installation (GI)* and *multiple Grid Installations (GIs)*. GIs refers to a computer installed with core Grid services from the GT4 toolkit and containing OGSA-DAI data service installed on top of the GT4 container. Hence each GI can be treated as a “representative” HealthGrid.

Although, for the purpose of experimentation, both these grid installations were confined to the physical location of the laboratory, the single GI installation typifies one operational HealthGrid and the multiple GIs represent operational HealthGrids situated at geographically distributed locations. Both single and multiple GIs contain one or more Data Sources (DSs), each of which contain large datasets. The relationship between single GI and multiple GIs, in relation to DSs, are shown below in Figure 43 and Figure 46 respectively. It is worth mentioning here that all the GIs used in both of these experiments contained semantically heterogeneous Data Sources. This heterogeneity was on the basis of semantically different data fields (attributes).

Two experiments were conducted, (a) *Experiment-I* for testing the system with one GI and large number of DSs, and (b) *Experiment-II* for testing the system with multiple GIs, each having one DS. Both these experiments had different network setups as shown in Table 5 and Table 8 respectively.

In order to meet the functional requirements of both the experiments, all the necessary installations were done in a fashion similar to the installations of the experimental prototype (Chapter 7). Therefore, core Grid services from the GT4 toolkit, OGSA-DAI (OGSA-DAI WSRF 2.2) and Postgres 8.2 were installed on all machines. For both of these experiments, in addition to MySQL another relational database, Postgres 8.2 was used which is also an open-source. This was done in order to test the system with a different Database. However, for *Experiment-II* other software components, namely, JAVA (JDK1.5.0\_09), Eclipse 3.2.2 and Apache Tomcat Server (version 5.0.28), were installed only on one machine, on which the client application code was running, as it was the application server GI from which the code was run and all the queries were sent.

The details of the hardware and software configurations used for both the experiments are listed in Table 4:

**Table 4: Hardware and Software Specifications**

SPECIFICATIONS	
<b>HARDWARE</b>	
CPU	Intel® Pentium® M Processor 1.73GHz
Memory	1 GB
<b>SOFTWARE</b>	
OS	Windows® XP
Globus Toolkit	GT4.0 (Core Web Services)
OGSA-DAI	OGSA-DAI WSRF 2.2
JAVA	Sun JDK1.5.0_09
Software Development Kit	Eclipse SDK 3.2.2
Application Server	Apache Tomcat Server (version 5.0.28)
Database	Postgres 8.2 / MySQL

After all necessary installations, the experiments were performed to test the HealthGrid prototype implementation. User queries were submitted through the application's Graphical User Interface (GUI), which is a JSP page (Chapter 7) and constitutes Component-III (WUI) of the proposed ASIDS architecture (Chapter 6). Examples of the SQL user queries used for these experiments are shown in Figure 41. 25 user queries were sent for both the experiments and each time a different query was submitted by changing the user keywords or user selection from the menu list on the JSP page. The results were recorded and at the end an average of the results was taken. Each of these experiments is described in detail further in this section.

**QUERY 1:**

```
SELECT * FROM (TABLE1, TABLE2, TABLE3, TABLE4, TABLE5, TABLE6, TABLE7,
TABLE8, TABLE9, TABLE10)
WHERE (COLUMN NAME1, COLUMN NAME2, COLUMN NAME3, COLUMN
NAME4, COLUMN NAME5, COLUMN NAME6, COLUMN NAME7, COLUMN
NAME8, COLUMN NAME9, COLUMN NAME10) = 'USER KEYWORD';
```

**QUERY 2:**

```
SELECT * FROM (TABLE1, TABLE2, TABLE3, TABLE4, TABLE5, TABLE6, TABLE7,
TABLE8, TABLE9, TABLE10)
WHERE (COLUMN NAME1, COLUMN NAME2, COLUMN NAME3, COLUMN
NAME4, COLUMN NAME5, COLUMN NAME6, COLUMN NAME7, COLUMN
NAME8, COLUMN NAME9, COLUMN NAME10) LIKE '%USER KEYWORD';
```

**QUERY 3:**

```
SELECT * FROM (TABLE1, TABLE2, TABLE3, TABLE4, TABLE5, TABLE6, TABLE7,
TABLE8, TABLE9, TABLE10)
WHERE (COLUMN NAME1, COLUMN NAME2, COLUMN NAME3, COLUMN
NAME4, COLUMN NAME5, COLUMN NAME6, COLUMN NAME7, COLUMN
NAME8, COLUMN NAME9, COLUMN NAME10) LIKE 'USER KEYWORD'%;
```

**Figure 41: Examples of SQL User Queries Used**



## 8.2 Experiment-I: Testing out the System with a Single Grid Installation

### 8.2.1 Overview of Experiment-I

In this experiment, implementation of the proposed ASIDS architecture on the HealthGrid prototype was tested with single GI containing multiple DSs (Figure 42), each of which had large datasets (Figure 43), to test if the proposed semantic interoperability approach is feasible with large datasets having semantically different data fields (attributes).

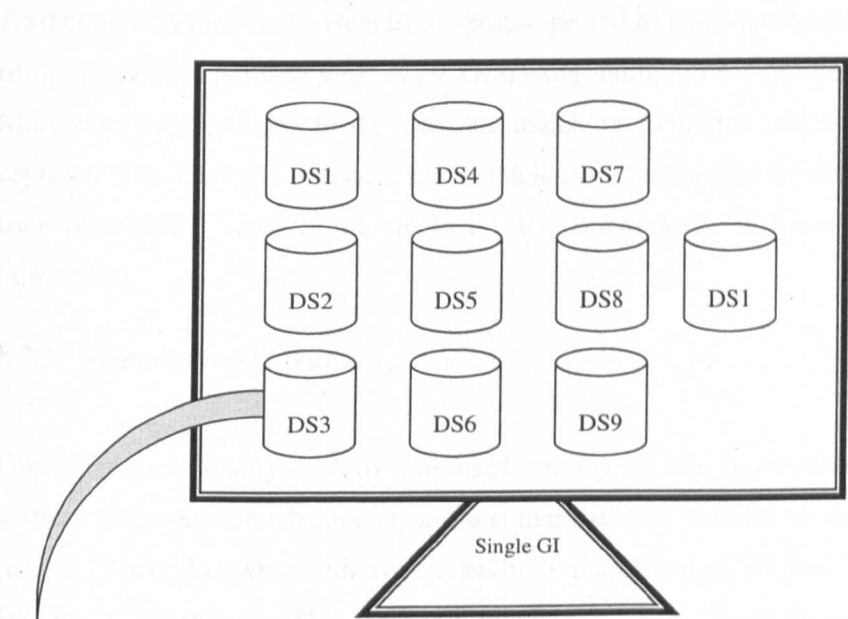


Figure 42: Single GI Containing Multiple DSs

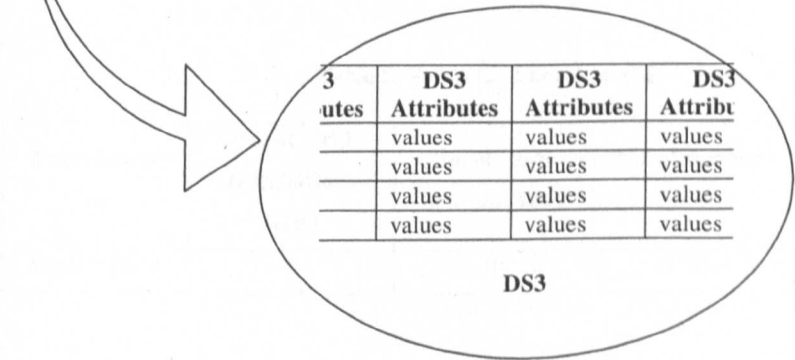


Figure 43: Data Sets in DS3

In order to conduct this test, only a single GI was used containing ten DSs with a total of 25,000 records (setup for *Experiment-I* is shown in Table 5). Different user queries were submitted to fetch the matching records to see if the system fetches data from all data sources regardless of their semantic heterogeneity. The elapsed time for fetching the matching records was noted for each of 25 runs and an average was taken at the end.

8.2.2 Objective of Experiment-I

The objective of conducting this experiment was to test the implementation of the ASIDS architecture on the HealthGrid prototype and to demonstrate the feasibility of the proposed approach with *single GI* having multiple DSs and large datasets. Moreover, overhead due to the semantic matching technique was needed to be captured. The reader is reminded that all datasets contained in the different DSs, each of which is used by the single GI, had semantically different data fields (attributes).

8.2.3 Experiment-I Setup

For Experiment-I only one GI was used, similar to the application prototype network (described in Chapter 7), and it contained large datasets of about 25,000 records. These 25,000 records were equally divided between 10 DSs. Thus, each DS has 2,500 records. This is shown in table 2 below. Since this experiment involved a single GI therefore all the 10 DSs were created on a single computer.

Table 5: Setup for Experiment-I

Experiment No.	No. of Grid Installations (GIs)	No. of Data Sources (DSs)	No. of Records per DS	Total No. of Records
Experiment-I	1	10	2,500	25,000

### 8.2.4 Results Analysis of Experiment-I

Different user queries were submitted to fetch the matching records. This difference was made by changing the keyword and menu item combinations from the JSP page. The reason for sending different queries was to see if the system fetches data from all data sources regardless of their semantic heterogeneity. The elapsed time for fetching the matching records was noted for each single DS for each run and an average of results was taken at the end. The time was monitored merely for checking the implementation of the system and the motive was not to improve the performance of the system as performance improvement is beyond the scope of this research.

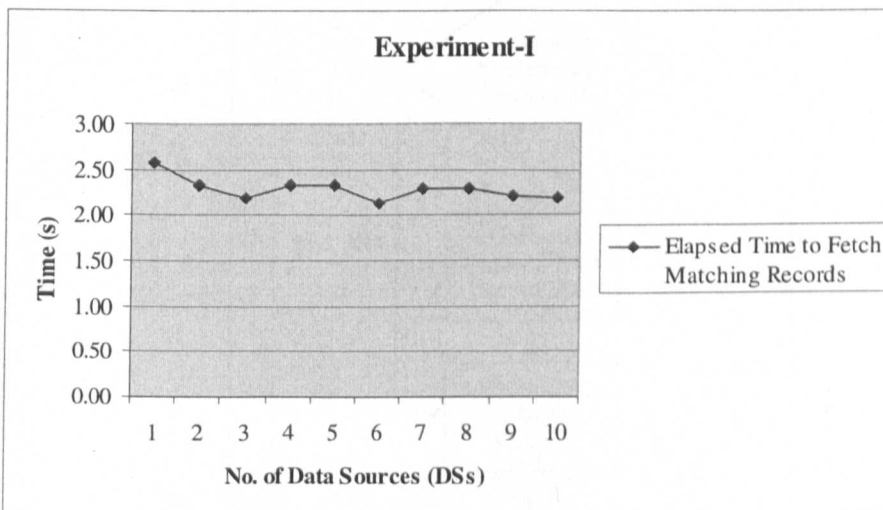


Figure 44: Testing out the System with Single Grid Installation

Figure 44 shows results from *Experiment-I*, plotted on a graph. On the X-axis is the number of data sources and on Y-axis is the time in seconds. The readings (points) shown on the graph represent the time it took to fetch matching records for each of the 10 DSs. However, the first reading not only includes the elapsed time to fetch matching records but also the time to generate a dynamic semantic map for all the 10 DSs. This dynamic semantic map generation has already been explained in Chapter 7. In order to understand how these readings for the elapsed

time have been obtained, it is important to know how the program works as shown in the flowchart (Figure 45).

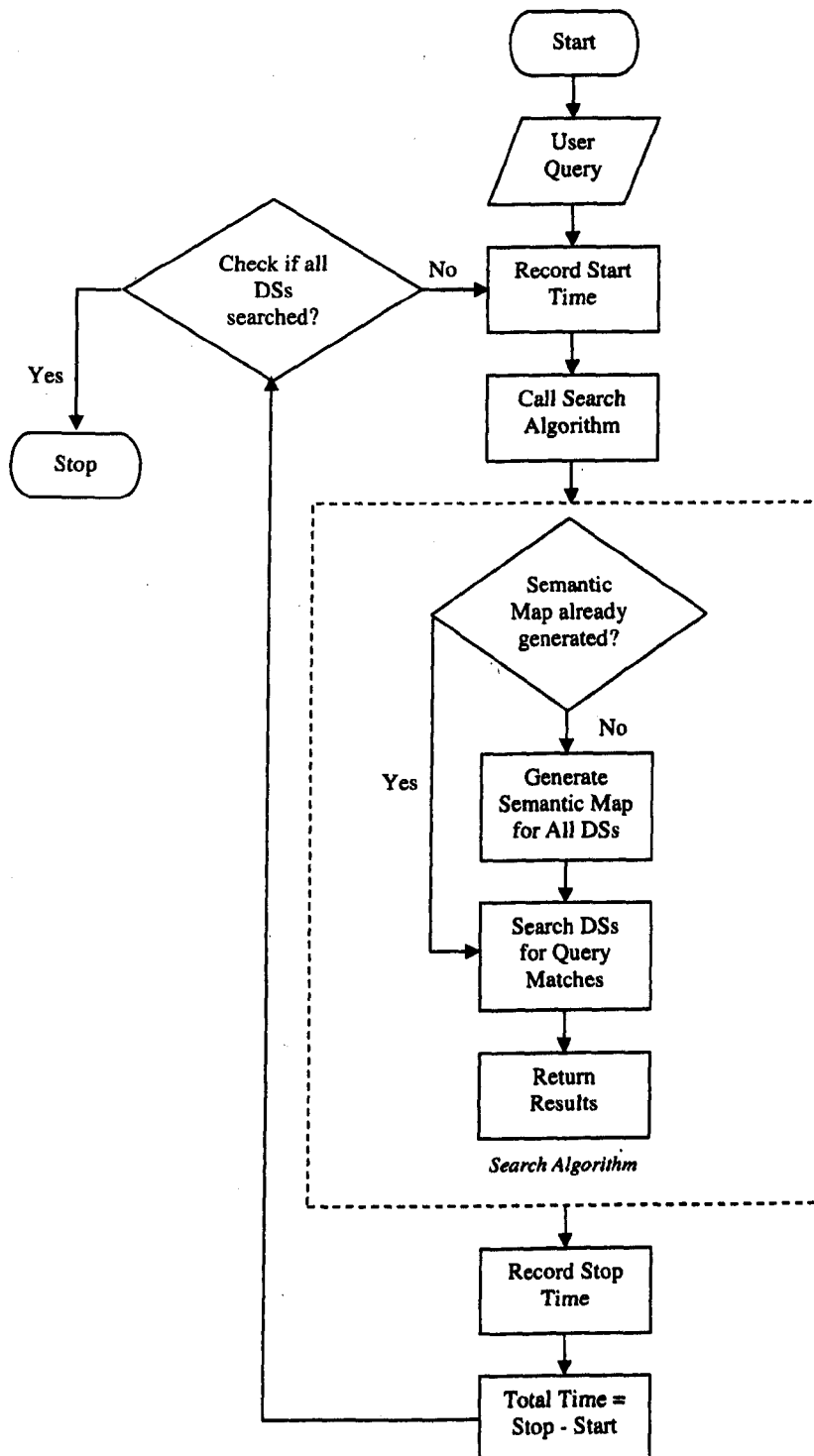


Figure 45: Flowchart for Experiment-I

It can be clearly seen from the flowchart that the semantic map for all the DSs is generated only once (for the first time) when the query is sent, and every time it is checked if the semantic map already exists.

The time taken to fetch matching records from each of the data sources can be seen from Table 6. For example the first data source was searched in 2.57 seconds; however this time also includes the time to generate a dynamic semantic map for all the 10 DSs, the second data source was searched in 2.32 seconds and so on in order to find the matching records. It can be seen from the graph that the results are showing a relatively linear pattern throughout the 10 DSs queried.

**Table 6: Elapsed Time to Query Single GI with Multiple DSs**

<b>Data Sources (DSs)</b>	<b>Elapsed Time to Fetch Records (seconds)</b>	<b>Cumulative Elapsed Time (seconds)</b>	<b>Records Matched</b>
DS1	2.57 (includes time to generate semantic map)	2.57	500
DS2	2.32	4.89	500
DS3	2.18	7.08	500
DS4	2.34	9.41	500
DS5	2.33	11.75	500
DS6	2.12	13.87	500
DS7	2.29	16.16	500
DS8	2.30	18.46	500
DS9	2.21	20.67	500
DS10	2.19	22.86	500
<b>TOTAL</b>	<b>22.86</b>		<b>5000</b>

As shown from the flowchart in Figure 45 and Table 6 above, the elapsed time for fetching records from DS1 also includes the time taken to generate the dynamic semantic map, whereas the elapsed time for fetching records from other DSs (2-10) does not include the time taken for generating the dynamic semantic map. Therefore, an average of elapsed times for DS2 to DS10 was taken and subtracted from the elapsed time for DS1 as shown in Table 7 to calculate the overhead of the proposed semantic approach which is negligible (0.32 seconds).

Table 7: Showing Overhead for Semantic Map Generation

Elapsed Time for DS1	Avg. Elapsed Time (DS2-DS10)	Semantic Map Generation Overhead
2.57	2.25	0.32

The table above shows that 0.32 seconds were required by the system to generate a semantic dynamic map. As this semantic map is generated at run time, therefore performance drops because first the map is generated and then the query is sent to the respective data sources. However, the elapsed time increment remains relatively flat (and is not much influenced) when increasing the number of records/rows because this delay is directly proportional to the number of GIs and not to the number of DSs. It is reasonable to argue that this overhead can be insignificant or just tolerable, when considering the utility and importance of the semantic search.

This test was conducted for testing out the system implementation of the proposed ASIDS architecture on the HealthGrid prototype with a single Grid Installation having multiple data sources and large data sets. It was shown that the system is functional with large data sets. The limitation of this experiment was the sequential or serial running of the search algorithm which caused unnecessary delays in the elapsed time. Moreover, running the algorithm in parallel or distributed manner instead of the serial or sequential manner would improve the performance of the system and can be included in the future research. Nevertheless, in case of larger numbers of records, the expectation is that the architecture would still be manageable, reusable and flexible, considering a relatively stable increment trend from the graph.

### 8.3 Experiment-II: Testing out the System with Multiple Grid Installations

#### 8.3.1 Overview of Experiment-II

In this experiment, implementation of the proposed ASIDS architecture on the HealthGrid prototype was tested with multiple geographically distributed Grid Installations to check if the system works on adding more number of Grid Installations/computers have multiple data sources with semantically different data fields (attributes) thus verifying that the proposed approach is practically applicable.

In order to conduct this test, multiple GIs were used containing one DS each (as shown in Figure 46), each of which had 2,500 records which makes a total of 12,500 records (setup for Experiment-II is shown in Table 8).

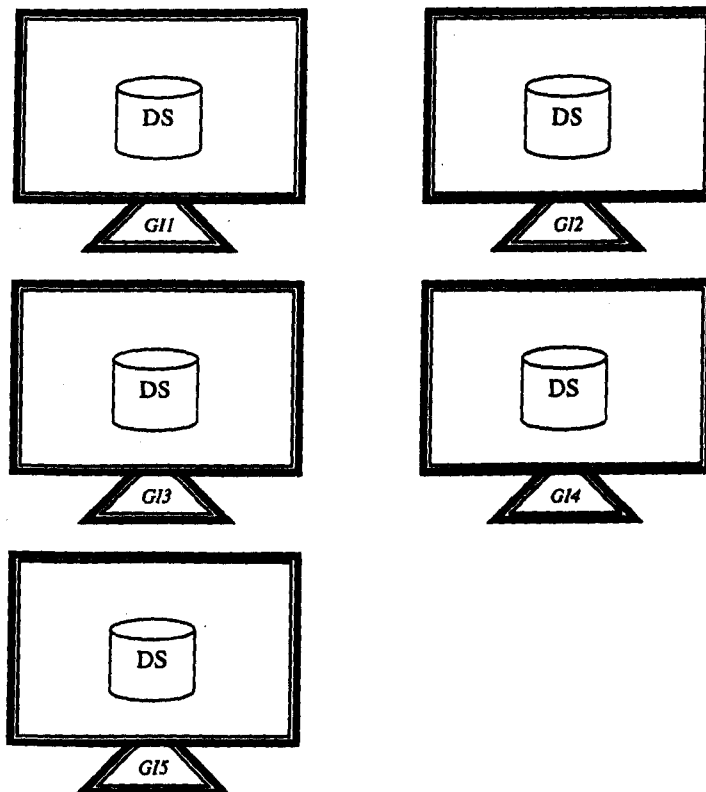


Figure 46: Multiple GIs Containing One DS Each

These GIs were to mimic the geographically distributed HealthGrids as in the real world scenario HealthGrids are located in different places. Unlike Experiment-I, different user queries were submitted to multiple DSs in multiple GIs to fetch matching records. This was done in order to see if the system fetches data from all semantically heterogeneous DSs contained in different GIs regardless of their semantic heterogeneity and distributed geographical locations. Elapsed time for fetching the matching records was noted for each run and an average was taken at the end.

### **8.3.2 Objective of Experiment-II**

The objective of conducting these experiments was to test implementation of the ASIDS architecture on multiple HealthGrids distributed in different geographical locations. This was to demonstrate that the proposed approach is practically applicable to *multiple* Grid Installations (increasing number of Grid Installations) at geographically distributed locations. Moreover, similar to Experiment-I, the overhead due to the semantic matching technique was needed to be captured which is shown in the results. All the GIs used in these experiments contained DSs that had semantically different data fields (attributes).

### **8.3.3 Experiment-II Setup**

For Experiment-II, the application prototype network (described in Chapter 7) was expanded to five Grid Installations (GIs) that were geographically distributed in order to conduct extensive experiments. For this purpose, up to 5 Grid Installations were used containing one DS in each GI and each of these data sources contained 2500 records or rows (as shown in Table 8). These DSs were the same as used for Experiment-I.



**Table 8: Setup for Experiment-II**

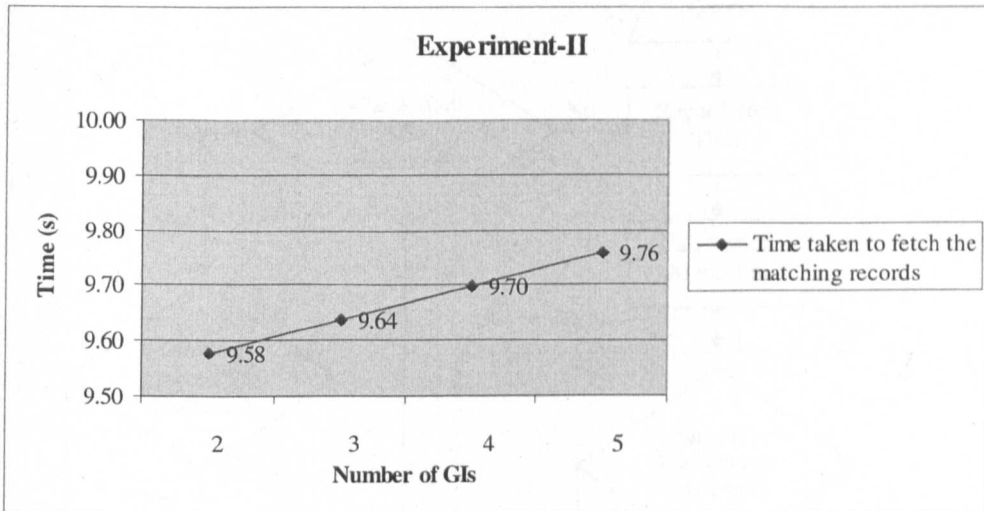
<b>Experiment No.</b>	<b>No. of Grid Installations (GIs)</b>	<b>No. of DSs in each GI</b>	<b>Total No. of Data Sources (DSs)</b>	<b>No. of Records per DS or GI</b>	<b>Total No. of Records</b>
Experiment-II	5	1	5	2,500	12,500

All of these 5 Grid Installations were setup to have similar hardware specifications but contained different or heterogeneous data sources i.e. they had semantically different data fields (attributes). This is because there are different HealthGrids in the real world that are geographically distributed and have semantically different DSs or datasets.

### **8.3.4 Results Analysis of Experiment-II**

For each number of Grid Installations (1-5), different queries were submitted and the time it took to fetch the matching rows was recorded. Similar to Experiment-I, the queries were different due to the changing the keyword and menu item combinations input through the JSP page which is an interface to the system. Unlike Experiment-I, the reason for sending different queries was to see if the system is capable of fetching data from geographically distributed HealthGrids or GIs regardless of their semantic heterogeneity. The experiment ran first time with only GI1 in the network setup, second time with GI1 and GI2 (after adding GI2 to the network setup), third time with GI1, GI2 and GI3 (after adding GI3 to the network setup) and so on until all the GIs (1-5) were added to the network. This was to test the application of the proposed approach (system) to increasing number of GIs. The elapsed time for fetching the matching records was noted for each run and an average of results was taken at the end. Here also, the time was monitored merely for checking the implementation of the system for multiple distributed GIs and not to improve the overall performance of the system, it is beyond the scope of this research and could be included in future research. Figure 47 shows the results from this experiment on a graph. On X-axis is the number of Grid Installations (GIs) and on Y-axis is the elapsed time (in seconds) until the

end of retrieval. The reader is reminded that, for this experiment, 1 GI contains 1 DS.



**Figure 47: Testing out the System with Multiple Grid Installations**

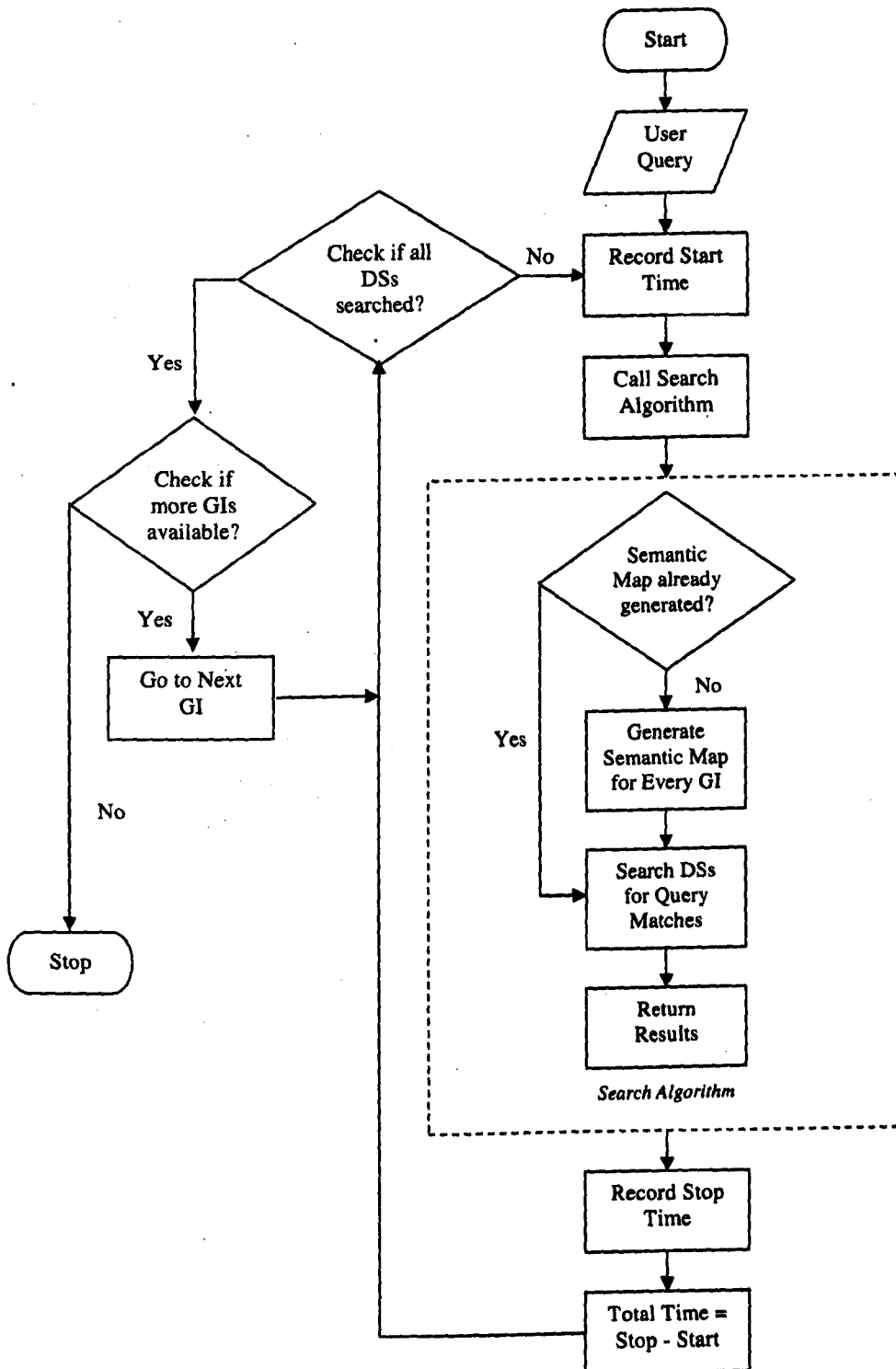


Figure 48: Flowchart for Experiment-II

It can be clearly seen from the flowchart (Figure 48) that the semantic map for every GI is generated only once (for the first time) when the query is sent, and every time it is checked if the semantic map, for that particular GI, already exists. The first Grid Installation (GI1) has not been included in the graph, as it was an outlier (very low value). As it can be seen from Table 9, it took 2.39 seconds to fetch 1000 matching records from GI1, which is quite different from the rest of the values from GI2-GI5. This low value for GI1 was obtained because it was the server Grid Installation (running Tomcat server) from which all the queries were sent and there was no network overhead, whereas for the other GIs (2-5) there was a network overhead. Therefore, as a special case GI1 was excluded from the comparison.

**Table 9: Elapsed Time to Query Multiple GIs Containing One DS Each**

<b>Grid Installation</b>	<b>Elapsed Time to Fetch Records and to Generate Semantic Map (seconds)</b>	<b>Cumulative Elapsed Time (seconds)</b>	<b>Records Matched</b>
GI1	2.39	2.39	1000
GI2	9.58	11.97	1000
GI3	9.64	21.61	1000
GI4	9.70	31.31	1000
GI5	9.76	41.07	1000
<b>TOTAL</b>	<b>41.07</b>		<b>5000</b>

GI2, GI3, GI4 and GI5 took 9.58, 9.64, 9.70 and 9.76 seconds respectively to fetch 1000 records that matched the query out of 5000 records (for each GI). It can be seen from Table 9 that the elapsed time includes both time to fetch the matching record and time to generate dynamic semantic map for each GI.

As shown from the flowchart in Figure 48 and Table 9 above, the elapsed time for fetching records from all the GIs (1-5) also includes the time taken to generate the dynamic semantic map. The total elapsed time calculation for each of the GIs in Experiment-II is similar to that of Experiment-I. Therefore, it can be clearly seen from Figure 49 that each GI is playing the same role as that of a single GI in

Experiment-I. As the semantic map is generated for each GI at the run time, therefore the start time and stop time is calculated for every GI independently.

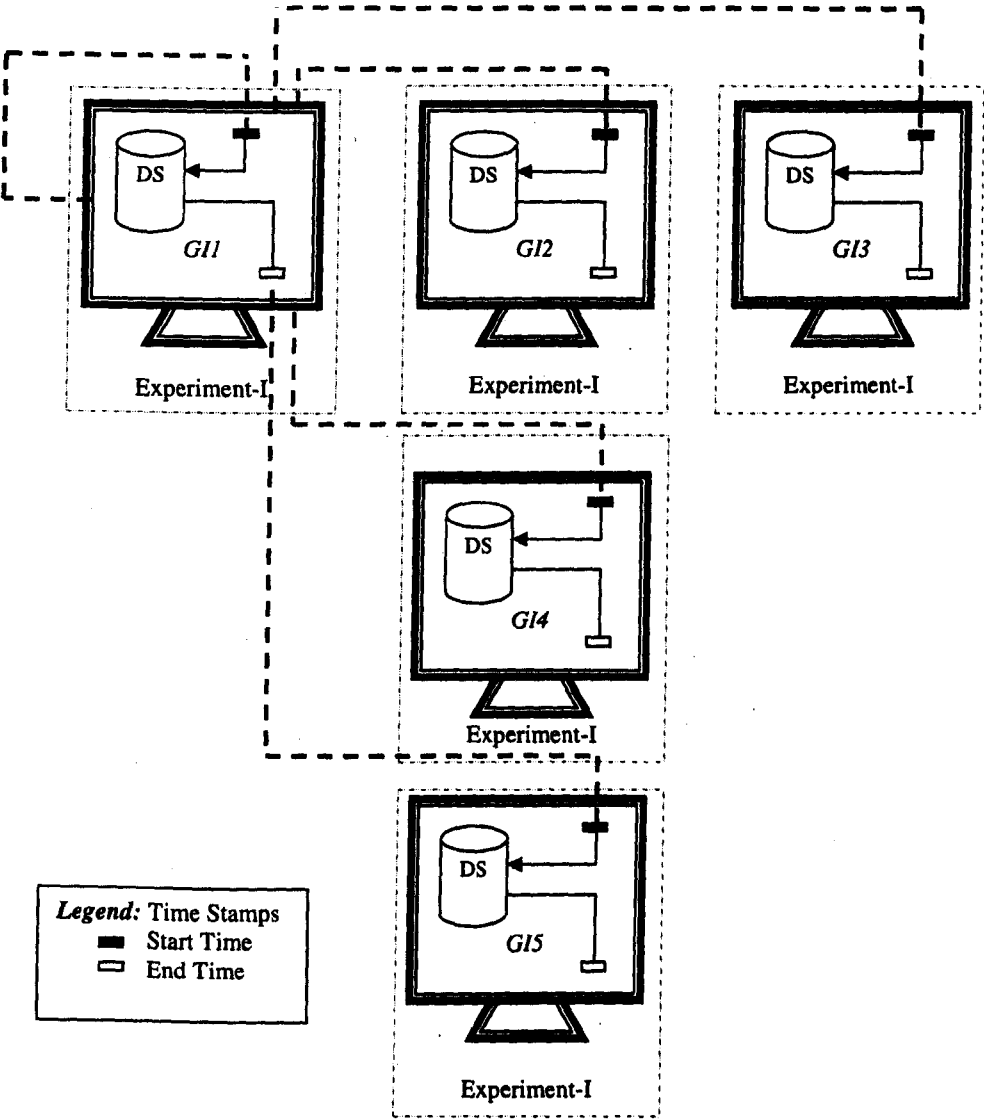


Figure 49: Time Stamps for Experiment-II

This test was conducted for testing out the system implementation of the proposed ASIDS architecture on the HealthGrid prototype with multiple geographically distributed Grid Installations having semantically different data fields (attributes). It was shown that the system (proposed ASIDS architecture) remains functional

with increasing number of Grid Installations/computers. It was shown that the system is functional with large data sets.

In case of larger numbers of Grid Installations and data sources, the expectation is that the architecture would still be manageable, reusable and flexible considering a relatively stable increment trend from the graph. The limitation of this experiment was same as that of Experiment-I, the sequential or serial running of the search algorithm that caused unnecessary delays in the elapsed time. If the search algorithm was run in parallel or in distributed manner then this would have saved an overhead of about 0.06 seconds for each GI.

## 8.4 Conclusions

This chapter aimed at testing out the proposed system, i.e. implementation of the ASIDS architecture on the HealthGrid prototype. Practical application of the proposed approach was shown both with single GI having large datasets of semantically different data fields or attributes and with increasing number of GIs (multiple GIs containing semantically different data fields or attributes).

For this purpose two different experiments were conducted. Experiment-I was for testing out the system with large datasets (having semantically different data fields) and second extensive experiment was for checking if the system works on adding more number of geographically distributed GIs (having semantically different data fields). For this reason both the experiments were conducted in different network setups. The elapsed time measurements were taken and results were plotted on the graphs. Results showed that the proposed semantic integration approach (ASIDS architecture) remains functional in both experiments. However, graphs show that there is a minor overhead. The reason for this overhead/delay is due to the fact that it takes time to generate a semantic dynamic map. As this semantic map is generated at the run time, therefore the performance drops because first, the map is generated, and then the query is sent to the respective DSs. The elapsed time increment remains relatively flat when increasing the number of records. It is reasonable to argue that this overhead can be insignificant

or just tolerable, when considering the utility and importance of the semantic search.

Moreover, it is expected that the architecture would still be manageable, reusable and flexible in case of even larger numbers of GIs and even increasing DSs just by making minor changes to the system configurations. Thus by following the proposed approach the contemporary Grid technologies could be used for integrating heterogeneous data resources that have semantically different data fields (attributes).

## Chapter 9

### Conclusions

#### 9.1 Introduction

This chapter provides a summary of the thesis and the conclusions from the research carried out. Moreover, avenues for future research are discussed, along with the extent to which these technologies would be applicable and adaptable for wider implementation. Finally, some reflections on this research are offered.

#### 9.2 Research Summary

Grid technologies have now been around for quite sometime and have been explored to some degree in meeting today's increasing information and communication demands. A need was identified to provide taxonomies of Grid resources and existing resource discovery methods, and a comprehensive literature review was conducted for this purpose. This achieves the *first* objective of this thesis. An investigation into the literature showed that their potential has been explored in a number of different ways and to resolve various resource discovery problems. Successful allocation, discovery, sharing and integration of Grid resources are issues not yet resolved. The data-type resources over the Grids are faced with a lot more challenges as compared to the other Grid resources because of their consistency, integrity and homogeneity constraints, especially if the nature of data is sensitive in terms of its application domain; for example, health-related data has to be dealt with even more care because it is life sensitive. In order to enable the global sharing of the data-type resources on Grids, it is important to integrate them in some way. It has been shown through the literature surveys and the technology analysis that the mainstream Grid technologies, such as GT4 and especially OGSA-DAI, have been employed to address the data



integration problems. However, this integration has not been achieved fully on a semantic basis, which is a key challenge for today's Grid community. By conducting a technology analysis the *third* objective of this thesis was accomplished. In this thesis, the problem of semantic integration of heterogeneous data resources has been explored in detail. On the basis of the conclusions drawn from the literature surveys and the technology analysis, the hypothesis of this thesis was formulated which states: *"Existing mainstream Grid technologies are sufficient for providing effective and sustainable solutions to the problem of semantically federating networked (heterogeneous) data resources."*

Hence the aim of this research was defined, which is, *to explore the possibility of using the mainstream Grid technologies to semantically integrate heterogeneous data sources in an effective, efficient and user-friendly way*. In order to test the above mentioned hypothesis, an Architecture to Semantically Integrate Data Sources (ASIDS) was proposed, which realised the *fourth* objective of this thesis.

A HealthGrid application prototype (ASIDSApplication) was build in Java and the ASIDS architecture was implemented in this application, to demonstrate the feasibility of semantically integrating heterogeneous data sources, this accomplishes the *fifth* objectives of this thesis, respectively. Since the proposed architecture was implemented on a HealthGrid exemplar, therefore there was a need to classify the various types of HealthGrids in order to produce a taxonomy and to see the need for semantic data integrity on HealthGrids. For this purpose, a comprehensive literature review of resource discovery in HealthGrids was conducted for producing a taxonomy and to examine the need for semantic data integrity on HealthGrids, this achieves the *second* objective of the thesis.

Finally, experiments were conducted to evaluate implementation of the proposed ASIDS architecture on the HealthGrid prototype and demonstrate feasibility of the proposed approach. Two experiments were conducted with different network setups. Evaluation results showed that the proposed semantic integration approach

(ASIDS architecture) is feasible and remains functional in both experiments. This accomplishes the *sixth* objective of this thesis.

*"How to facilitate the semantic federation of heterogeneous data resources using mainstream Grid technologies?"*

### 9.3 Conclusions

The detailed literature surveys concluded formed the basis of the research question and thus the hypothesis was highlighted. It was learnt that the semi-distributed resource discovery model can provide the best option for creating request brokering systems and designing related middleware packages for discovering the resources on Grids, since overall it seems to be more reliable. However, this model also has some limitations, such as complexity, time, costs and difficulty of managing or maintaining. In order to provide optimal service, such systems need to be easily configurable (manageable), flexible and generic (reusable). Moreover, a semi-distributed network architecture should be modelled in a sophisticated manner, so as to address the scalability issue sufficiently to ensure that its effectiveness and efficiency remain unaltered regardless of the number of nodes or peers or resources added or removed from the network. Moreover, it seemed that using a Hybrid approach over a Semi-Distributed architectural model can help resolve the problem of resource discovery in Grids to some extent.

Through a review of the current implementations of HealthGrids, it was learnt that there is a case for using Grid technology in healthcare that arose mainly from the need to improve, safeguard and effectively exploit the available *life-significant medical information*, the need to protect the privacy of *personal, life-sensitive health information*, and the need to provide *integrated healthcare services* and have in place effective, global *channels of collaboration*. For the long-term future, there is a need for the various Grid-enabled applications to be designed specifically for HealthGrids that also serve as an effective channel for international collaborations. It was also learnt that to exploit effectively the wealth

of medical information, there was an urgent need to integrate, manipulate, process, and analyse huge heterogeneous datasets from disparate sources. More systematic use of Grid technology in healthcare will not only help meet the current needs for data processing, but will ensure that future demand for even more capacity to deal with far larger volumes of data can be met.

A critical question arose as to how metadata can support the integration of two or more heterogeneous objects as there is also a need to have a semantic integration of various resources that are geographically or organisationally spread, so that they can be shared and utilized globally on a HealthGrid. The emergent semantic networks ensure the integrity of meaning between different concepts and can play an important role in solving this complex integration problem. Moreover, it has been witnessed from the literature survey that the mainstream Grid technologies such as OGSA-DAI can prove to be a candidate solution to the data federation problem.

A technology analysis of the mainstream Grid and Web technologies suggested that some of them, mainly GT4 and OGSA-DAI, can provide candidate solutions to the semantic data integrity issue and this paved our way to the proposed architecture. It seemed that to in order to address the research question, the Empirical approach would be well suited as this research involved implementation of the proposed ASIDS architecture within an exemplary ASIDSApplication environment (the HealthGrid exemplar).

Since using a Hybrid approach over a Semi-Distributed architectural model was highlighted through the literature surveys, therefore keeping this fact into consideration, an n-tier-to-n-tier application architecture (ASIDS), for semantically integrating heterogeneous data resources, was proposed. It followed the Semi-Distributed architectural model therefore, its design was expanded to more than one tiers (or n-tiers). Moreover, the Hybrid approach (by combining the P2P and Semantic approaches), was used in a sophisticated manner (as can be

seen from the Component-I, II and III of the ASIDS design Figure 19), to provide an optimal solution to the research problem.

This architecture provided a basis for the feasible implementation of the experimental prototype and was implemented by building a functional prototype in an exemplar HealthGrid environment in order to validate the hypothesis. The ASIDSApplication prototype was built and set up according to the ASIDS architecture using the mainstream Grid technologies. When the user query was sent, it fetched data from all the heterogeneous fields of physically distributed, heterogeneous data sources, resolving the interoperability issue and semantically federating the heterogeneous data resources, thus verifying the hypothesis of this research.

The proposed ASIDS architecture was evaluated in terms of two evaluation analyses experiments, one for testing out the system with large datasets and second for checking if the system works on adding more number of geographically distributed nodes. Results from the evaluation experiments showed that the proposed semantic integration approach (ASIDS architecture) is reliable and stays operational even with larger number of nodes and/or records and it could be easily scaled out to add more nodes just by making minor changes to the configurations. However, as the architecture is flexible, manageable and reusable, it is expected to provide an optimal service (as is expected from systems using the Hybrid approach over a Semi-Distributed architectural model). Thus by following the proposed approach the contemporary Grid technologies could be used for integrating heterogeneous data resources that have semantically different data fields (attributes).

It has been learnt that the issue of semantically federating or integrating heterogeneous data sources was addressed by using the mainstream Grid technologies, an ASIDS architecture was proposed and implemented on an exemplar HealthGrids prototype (ASIDSApplication) that fetched data from all the heterogeneous fields of physically distributed, heterogeneous data sources,

resolving the interoperability issue and semantically federating the heterogeneous data resources, thus verifying the hypothesis of this research.

The research would be beneficial for scientific collaboration and group-wise analysis operations where data integration is a necessity. As the research has been implemented on a pilot HealthGrids prototype (exemplar), it would be quite beneficial for the healthcare sector eventually promoting e-Health.

## **9.4 Further Research**

The contribution of this thesis is an approach that uses contemporary Grid technologies for integrating heterogeneous data resources that have semantically different data fields (attributes). The approach is demonstrated using a prototype HealthGrid. The proposed approach that leads to the ASIDS architecture is novel as it performs semantic matching at the data field-level or attribute-level and without using any of the complex industry-developed semantic mapping tools, which is the unique characteristic of ASIDS. Thus the novelty, significance and usefulness of the proposed rational approach is that it provides a simple pragmatic solution to the extremely difficult and complex problem of semantic integration and interoperability of heterogeneous data resources in Grids. Moreover, without significant extra effort, this approach can be applied to address syntactic heterogeneity.

This study lays the grounds for future research that could be carried based on the lessons learnt from this study. There are many avenues for the future research that can be explored on the basis of this study. The technologies deployed could prove to function to a better extent and can be applicable and adaptable to be adopted more widely.

## References

- Afgan, E. (2004). Role of the resource broker in the Grid. *In Proceedings of the 42nd annual Southeast regional conference*, pp. 299-300. ACM Press, New York, NY.
- Aloisio, G., Cafaro, M., Fiore, S. and Mirto, M. (2005a). ProGenGrid: a grid-enabled platform for bioinformatics. *Studies in Health Technology and Informatics*, 112: 113-126.
- Aloisio, G., Cafaro, M., Fiore, S. and Mirto, M. (2005b). ProGenGrid: a workflow service infrastructure for composing and executing bioinformatics grid services. *In Proceedings of the 18th IEEE symposium on computer-based medical systems*, pp. 555-560. IEEE Computer Society, Washington, DC, USA.
- Alonso, J.M., Hernandez, V. and Molto, G. (2005). Experiences on a large scale grid deployment with a computationally intensive biomedical application. *In Proceedings of the 18th IEEE symposium on computer-based medical systems*, pp. 567-569. IEEE Computer Society, Washington, DC, USA.
- Andreozzi, S. (2004). GLUE Schema Implementation for the LDAP Data Model. Technical Report INFN/TC-04/16, INFN. Available online <http://www.cnaf.infn.it/~andreozzi/publications/Glue4LDAP.pdf>. Last accessed 23rd May 2007.
- Andrzejak, A. and Zhichen Xu. (2002). Scalable, efficient range queries for grid information services. *In Proceedings of the 2nd international conference on peer-to-peer computing*, pp. 33-40. IEEE Computer Society, Washington, DC, USA.
- Ankolenkar, A., Burstein, M., Cao Son, T., Hobbs, J., Lassila, O., Martin, D., McDermott, S., McIlraith, S. Narayanan, M. Paolucci, T. Payne, K. Sycara, and Zeng, H. (2001). DAML-S: Semantic Markup for Web Services. Available online <http://www.daml.org/services/daml-s/2001/10/daml-s.html>. Last accessed 23rd May 2007.

- Antonioletti, M., Atkinson, M., Baxter, R., Borley, A., Hong, N.P.C., Collins, B., Hardman, N., Hume, A.C., Knox, A., Jackson, M., Krause, A., Laws, S., Magowan, J., Paton, N.W., Pearson, D., Sugden, T., Watson, P. and Westhead, M. (2005). The design and implementation of grid database services in OGSA-DAI. *Concurrency Computation Practice and Experience*, 17(2-4): 357-376.
- Antonopoulos, N. and Salter, J. (2004). Efficient resource discovery in grids and P2P networks. *Internet Research*, 14(5): 339-346.
- Aversa, R., Di Martino, B., Mazzocca, N. and Venticinque, S. (2003). MAGDA: a software environment for mobile agent based distributed applications. *In Proceedings of the 11th Euromicro conference on parallel, distributed and network-based processing*, pp. 332-338. IEEE Computer Society, Washington, DC, USA.
- Aversa, R., Di Martino, B., Mazzocca, N. and Venticinque, S. (2004). Terminal-aware grid resource and service discovery and access based on agents technology. *In Proceedings of the 12th Euromicro conference on parallel, distributed and network-based processing*, pp. 40-45. IEEE Computer Society, Washington, DC, USA.
- Baker, M., Apon, A., Ferner, C. and Brown, J. (2005). Emerging grid standards. *Computer*, 38(4): 43-50.
- Benkner, S., Berti, G., Engelbrecht, G., Fingberg, J., Kohring, G., Middleton, S.E. and Schmidt, R. (2005). GEMSS: Grid-infrastructure for medical service provision. *Methods of Information in Medicine*, 44(2): 177-181.
- Berman, F., Chien, A., Cooper, K., Dongarra, J., Foster, I., Gannon, D., Johnsson, L., Kennedy, K., Kesselman, C., Mellor-Crumme, J., Reed, D., Torczon, L. and Wolski, R. (2001). The GrADS project: Software support for high-level grid application development. *International Journal of High Performance Computing Applications*, 15(4): 327-344.

- Berti, G., Benker, S., Fenner, J. W., Fingberg, J., Lonsdale, G., Middleton, S. E. and Surridge, M. (2003). Medical simulation services via the Grid. In *Proceedings of HealthGrid Workshop 2003*. Available online <http://www.ccr-l-nece.de/gemss/Presentations/healthgrid.pdf>. Last accessed 23rd May 2007.
- Bilykh, I., Bychkov, Y., Dahlem, D., Jahnke, J.H., McCallum, G., Obry, C., Onabajo, A. and Kuziemsky, C. (2003). Can grid services provide answers to the challenges of national health information sharing? In *Proceedings of the 2003 conference of the Centre for Advanced Studies on Collaborative research (CASCON '03)*, pp. 39-53. IBM Press, USA.
- Blake, J.A. and Bult, C.J. (2006). Beyond the data deluge: Data integration and bio-ontologies. *Journal of Biomedical Informatics*, 39(3): 314-320.
- Boccia, V., Guarracino, M.R., D'Amore, L. and Laccetti, G. (2005). A grid enabled PSE for medical imaging: experiences on MedIGrid. In *Proceedings of the 18th IEEE Symposium on computer-based medical systems*, pp. 529-536. IEEE Computer Society, Washington, DC, USA.
- Brady, J. M., Gavaghan, D. J., Simpson, A. C., Parada, M. M. and Highnam, R. P. (2003). eDiaMoND: A grid-enabled federated database of annotated mammograms. In *Berman, F., Fox, G. C. and Hey, A. J. C. (eds.), Grid computing: making the global infrastructure a reality*, pp. 923-943 (3rd edition). Hoboken, NJ: Wiley.
- Breton, V., Dean, K. and Solomonides, T. (2005). The healthgrid white paper. *Studies in Health Technology and Informatics*, 112: 249-321.
- Breton, V., Medina, R. and Montagnat, J. (2003). DataGrid, prototype of a Biomedical Grid. *Methods of Information in Medicine*, 42(2): 143-148.
- Brittain, J. and Darwin, I. F. (2003). *Tomcat: The Definitive Guide* (1st edition). UK: O'Reilly.



- Brooke, J., Fellows, D., Garwood, K. and Goble, C. (2004). Semantic Matching of Grid Resource Descriptions, pp. 240-249. In *Lecture notes in Computer Science*, volume 3165, Springer-Verlag, Germany.
- Buyya, R., Abramson, D. and Giddy, J. (2000). Nimrod/G: an architecture for a resource management and scheduling system in a global computational grid. In *Proceedings of the 4<sup>th</sup> international conference on High Performance Computing in the Asia-Pacific Region*, pp. 283-289. IEEE Computer Society, Washington, DC, USA.
- Buyya, R., Chapin, S. and DiNucci, D. (2000). Architectural Models for Resource Management in the Grid. In *Proceedings of the 1st international workshop on grid computing*. In *Lecture notes in Computer Science*, volume 1971, Springer-Verlag, Germany.
- Cannataro, M., Comito, C., Congiusta, A. and Veltri, P. (2004). PROTEUS: a bioinformatics problem solving environment on grids. *Parallel Processing Letters*, 14 (2): 217-237.
- Cannataro, M., Guzzi, P. H., Mazza, T., Tradigo, G. and Veltri, P. (2005). Preprocessing of mass spectrometry proteomics data on the grid. In *Proceedings of the 18th international symposium on computer-based medical systems*, pp.549-554. IEEE Computer Society, Washington, DC, USA.
- Chander, A., Dawson, S., Lincoln, P. and Stringer-Calvert, D. (2002). NEVRLATE: Scalable Resource Discovery. In *Proceedings of the 2nd international symposium on cluster computing*, pp. 382-388. IEEE Computer Society, Washington, DC, USA.
- Chapman, B.M., Sundaram, B. and Thygaraja, K. (2001). EZ-Grid System: Integrated Resource Brokerage services for Computational Grids. Available online <http://www2.cs.uh.edu/~ezgrid/EZ-GridAbstract.pdf>. Last accessed 27th March 2007.

- Chen, H., Jiang, J and Zhang, B. (2004). Design of an artificial-neural-network-based application-oriented grid resource discovery service. In *Proceedings of the 2004 international conference on machine learning and cybernetics*, pp. 2623-2628. IEEE Computer Society, Washington, DC, USA.
- Clarke, I., Sandberg, O., Wiley, B. and Hong, T.W. (2001). Freenet: A Distributed Anonymous Information Storage and Retrieval System. In *Proceedings of the international workshop on design issues in anonymity and unobservability*. In Lecture notes in Computer Science, volume 2009, Springer-Verlag, Germany.
- Coiera E. (2003). Guide to health informatics. London, UK: Hodder Arnold.
- Congiusta, A., Talia, D. and Trunfio, P. (2007). Distributed data mining services leveraging WSRF. *Future Generation Computer Systems*, 23(1): 34-41.
- Crespo, A. and Garcia-Molina, H. (2002). Routing indices for peer-to-peer systems. In *Proceedings of the 22nd International Conference on Distributed Computing Systems (ICDCS'02)*, pp. 23-32. IEEE Computer Society, Washington, DC, USA.
- Crompton, S. Y., Matthews, B. M., Gray, W. A., Jones, A. C., White, R. J. and Pahwa, J. S. (2006). OGSA-DAI and bioinformatics grids: Challenges, experience and strategies. In *Proceedings of the 6th IEEE International Symposium on Cluster Computing and the Grid (CCGRID 2006)*, pp. 193-200. IEEE Computer Society, Washington, DC, USA.
- CrossGrid.org. (2004). CrossGrid project homepage. Website <http://www.crossgrid.org/main.html>. Last accessed March 26, 2007.
- Czajkowski, K., Ferguson, D., Foster, I., Frey, J., Graham, S., Maguire, T., Snelling, D. and Tuecke, S. (2004a). From Open Grid Services Infrastructure to WS-Resource Framework: Refactoring & Evolution. The Globus Alliance. Available online [http://www.globus.org/wsrf/specs/ogsi\\_to\\_wsrf\\_1.0.pdf](http://www.globus.org/wsrf/specs/ogsi_to_wsrf_1.0.pdf). Last accessed 23rd May 2007.

- Czajkowski, K., Ferguson, D., Foster, I., Frey, J., Graham, S., Sedukhin, I., Snelling, D., Tuecke, S. and Vambenepe, W. (2004b). *The WS-Resource Framework*. The Globus Alliance. Available online <http://www.globus.org/wsrf/specs/ws-wsrf.pdf>. Last accessed 23rd May 2007.
- Czajkowski, K., Fitzgerald, S., Foster, I. and Kesselman, C. (2001). Grid information services for distributed resource sharing. In *Proceedings of the 10th IEEE International Symposium on High Performance Distributed Computing*, pp. 181-194. IEEE Computer Society, Washington, DC, USA.
- DARPA Agent Markup Language. (2007). Website <http://www.daml.org/>. Last accessed 25th May 2007.
- DataTag (2007). DataTag project description. Website <http://datatag.web.cern.ch/datatag/project.html>. Last accessed 23rd May 2007.
- David, B., Sergio, D.C. and Mark, L.E. (2005). Semantic Transformation of Web Services, pp. 856-865. In *Lecture notes in Computer Science*, volume 3762, Springer-Verlag, Germany.
- Doan, A., Madhavan, J., Domingos, P. and Halevy, A. (2002). Learning to map between ontologies on the semantic web. In *Proceedings of the 11th International Conference on World Wide Web*, Hawaii, USA, pp. 662-673.
- Dolin, R. H., Alschuler, L., Beebe, C., Biron, P. V., Boyer, S. L., Essin, D., Kimber, E., Lincoln, T. and Mattison, J. E. (2001). The HL7 Clinical Document Architecture. *Journal of the American Medical Informatics Association*, 8(6): 552-569.
- EGEE. (2007). Enabling grids for e-science project. Website <http://www.eu-egee.org/>. Last accessed 23rd May 2007.
- Ellisman, M., Brady, M., Hart, D., Lin, F., Muller, M. and Smarr, L. (2004). The emerging role of biogrids. *Communications of the ACM*, 47(11): 52-57.

- Ellisman, M. and Peltier, S. (2004). Medical data federation: the biomedical informatics research network. In Foster, I. and Kesselman, C. (eds.), *The Grid: Blueprint for a New Computing Infrastructure (2nd Edition)*, chapter 8. San Francisco, CA: Morgan Kaufmann.
- Erwin, D.W. and Snelling, D.F. (2001). UNICORE: A Grid Computing Environment. In *Proceedings of the 7th International Euro-Par Conference on Parallel Processing*. In Lecture notes in Computer Science, volume 2150, Springer-Verlag, Germany.
- Estrella, F., Hauer, T., McClatchey, R., Odeh, M., Rogulin, D. and Solomonides, T. Experiences of engineering Grid-based medical software (2007). *International Journal of Medical Informatics*, 76(8): 621-32.
- Fitzgerald, S., Foster, I., Kesselman, C., Von Laszewski, G., Smith, W. and Tuecke, S. (1997). A directory service for configuring high-performance distributed computations. In *Proceedings of the 6th international symposium on high performance distributed computing*, pp. 365-375. IEEE Computer Society, Washington, DC, USA.
- Foster, I., Berry, D., Djaoui, A., Grimshaw, A., Horn, B., Kishimoto, H., Maciel, F., Savva, A., Siebenlist, F., Subramaniam, R., Treadwell, J. and Reich, J. V. (2004). The Open Grid Services Architecture. Document number GFD-I.030. Available online <http://www.ogf.org/documents/GFD.30.pdf>. Last accessed 23rd May 2007.
- Foster, I., Czajkowski, K., Ferguson, D. E., Frey, J., Graham, S., Maguire, T., Snelling, D. and Tuecke, S. (2005). Modelling and managing state in distributed systems: the role of OGSF and WSRF. *Proceedings of the IEEE*, 93(3): 604-612.
- Foster, I. and Iamnitchi, A. (2003). On death, taxes, and the convergence of peer-to-peer and grid computing. In *Proceedings of the 2nd International Workshop on Peer-to-Peer Systems (IPTPS'03)*, pp.118-128. In Kaashoek, F. and Stoica, I.

- (eds.), Lecture notes in Computer Science, volume 2735, Springer-Verlag, Germany.
- Foster, I. and Kesselman, C. (1997). Globus: a Metacomputing Infrastructure Toolkit. *International Journal of High Performance Computing Applications*, 11(2): 115-128.
- Foster, I. and Kesselman, C. (1998). The grid: blueprint for a new computing infrastructure. San Francisco, CA: Morgan Kaufmann.
- Foster, I., Kesselman, C., Nick, J. M. and Tuecke, S. (2002). Grid services for distributed system integration. *IEEE Computer*, 35(6): 37-46.
- Foster, I., Kesselman, C., Nick, J. M. and Tuecke, S. (2003). The physiology of the grid, pp. 217-249. In Berman, F., Fox, G. and Hey, T. (eds.), *Grid Computing: Making the Global Infrastructure a Reality*. New York, NY: John Wiley & Sons.
- Foster, I., Kesselman, C. and Tuecke, S. (2001). The anatomy of the grid: enabling scalable virtual organizations. *International Journal of High Performance Computing Applications*, 15(3): 200-222.
- Foster, I. and Tuecke, S. (2005). Describing the elephant: the different faces of IT as service. *Queue*, 3(6): 26-29.
- Geddes, J., Lloyd, S., Simpson, A., Rossor, M., Fox, N., Hill, D., Hajnal, J. V., Lawrie, S., McIntosh, A., Johnstone, E., Wardlaw, J., Perry, D., Procter, R., Bath, P. and Bullmore, E. (2005). NeuroGrid: using grid technology to advance neuroscience. In *proceedings of the 18th IEEE symposium on computer-based medical systems*, pp. 570-572. IEEE Computer Society, Washington, DC, USA.
- Glatard, T., Montagnat, J. and Pennec, X. (2005). Grid-enabled workflows for data intensive medical applications, pp. 537-542. In *proceedings of the 18th IEEE symposium on computer-based medical systems*, pp. 570-572. IEEE Computer Society, Washington, DC, USA.

- Globus Alliance. (2007). Research Papers from Globus Alliance Members .Website <http://www.globus.org/alliance/publications/papers.php>. Last accessed 23rd May 2007.
- Globus Consortium Journal. (2006). Grid Analyst Round Up. Available online <http://www.globusconsortium.org/journal/20061102/index.php>. Last accessed 25th May 2007.
- Globus project. (2007). Website <http://www.globus.org/>. Last accessed 25th May 2007.
- Gonzalez-Velez, V. and Gonzalez-Velez, H. (2005). A grid-based stochastic simulation of unitary and membrane  $Ca^{2+}$  currents in spherical cells. In *proceedings of the 18th IEEE symposium on computer-based medical systems*, pp. 171-176. IEEE Computer Society, Washington, DC, USA.
- Harchol-Balter, M., Leighton, T. and Lewin, D. (1999). Resource discovery in distributed networks In *Proceedings of the 18th annual ACM symposium on Principles of distributed computing*, pp. 229-237. ACM Press, New York, NY, USA.
- Health Level Seven Inc. (2007). Health Level Seven homepage. Website <http://www.hl7.org/>. Last accessed 23rd March 2007.
- Heine, F., Hovestadt, M. and Kao, O. (2004). Towards ontology-driven P2P grid resource discovery. In *Proceedings of the 5th international workshop on grid computing*, pp.76-83. IEEE Computer Society, Washington, DC, USA.
- Houghton, J. (2002). Information Technology and the Revolution in Healthcare. Working paper 4, Centre for Strategic Economic Studies, Victoria University of Technology, Australia. Available online [http://www.cfses.com/documents/pharma/04-IT\\_and\\_Healthcare.PDF](http://www.cfses.com/documents/pharma/04-IT_and_Healthcare.PDF). Last accessed 23rd May 2007.

- Huang, X., Huang, L. and Li, M. (2006). *Grid-enabled medical image processing application system based on OGSA-DAI techniques*. In *Proceedings of the International workshops on advanced Web, network technologies and Applications*, pp. 460-464. In Shen, H. T., Li, J., Li, M., Ni, J. and Wang, W. (eds.), *Lecture notes in Computer Science*, volume 3842, Springer-Verlag, Germany.
- Hyatt, K. (2007). *N-Tier Application Development with Microsoft.NET*. Available online  
[www.microsoft.com/belux/msdn/nl/community/columns/hyatt/ntier1.aspx](http://www.microsoft.com/belux/msdn/nl/community/columns/hyatt/ntier1.aspx). Last accessed 23rd May 2007.
- Iamnitchi, A. and Foster, I. (2004). A peer-to-peer approach to resource location in Grid environments. In *Nabrzyski, J., Schopf, J. M. and Weglarz, J (eds.), Grid resource management: state of the art and future trends*, pp. 413-429. Norwell, MA, USA: Kluwer Academic Publishers.
- Iamnitchi, A. and Foster, I. (2002). On Fully Decentralized Resource Discovery in Grid Environments. In *proceedings of the 2nd international workshop on grid computing*, pp. 51-62. *Lecture notes in Computer Science*, volume 2242, Springer-Verlag, Germany.
- Iyengar, V., Tilak, S., Lewis, M. J. and Abu-Ghazaleh, N. B. (2004). Non-uniform information dissemination for dynamic grid resource discovery. In *Proceedings of the 3rd IEEE International Symposium on Network Computing and Applications*, pp. 97-106. IEEE Computer Society, Washington, DC, USA.
- Jeffery, K. G. (2007). Next Generation GRIDs for environmental science. *Environmental Modelling and Software*, 22(3): 281-287.
- Jensen, K. (1997). A brief introduction to coloured Petri Nets, pp. 203-208. In *Lecture notes in Computer Science*, volume 1217, Springer-Verlag, Germany.
- Jithesh, P. V., Kelly, N., Donachy, P., Harmer, T., Perrott, R., McCurley, M., Townsley, M., Johnston, J. and McKee, S. (2005). GeneGrid: grid based solution

for bioinformatics application integration and experiment execution. In *proceedings of the 18th IEEE symposium on computer-based medical systems*, pp. 523-528. IEEE Computer Society, Washington, DC, USA.

Johnson, C. (2003). What is research in computing science? Teaching notes, Department of Computer Science, University of Glasgow. Available online [http://www.dcs.gla.ac.uk/~johnson/teaching/research\\_skills/research.html](http://www.dcs.gla.ac.uk/~johnson/teaching/research_skills/research.html). Last accessed 23rd May 2007.

Johnson, W. and Brooke, J. (2002). Core Grid Functions: A Minimal Architecture for Grids. Presentation: Grid Protocol Architecture Working Group, Global Grid Forum. Available online [http://dsd.lbl.gov/~johnston/Grids/Minimal\\_Grid\\_Architecture.pdf](http://dsd.lbl.gov/~johnston/Grids/Minimal_Grid_Architecture.pdf). Last accessed 23rd May 2007.

Jones, D. M., Fenner, J. W., Berti, G., Kruggel, F., Mehrem, R. A., Backfrieder, W., Moore, R. and Geltmeier, A. (2004). The GEMSS Grid: An evolving HPC environment for medical applications. In *Proceedings of HealthGrid 2004*. Available online <http://www.ccrl-nece.de/gemss/Reports/Jones-healthgrid2004.pdf>. Last accessed 23rd May 2007.

Joseph, J., Ernest, M. and Fellenstein, C. (2004). Evolution of grid computing architecture and grid adoption models. *IBM Systems Journal*, 43(4): 624-645.

Jun, K., Boloni, L., Palacz, K. and Marinescu, D. C. (2000). Agent-based resource discovery. In *Proceedings of the 9th heterogeneous computing workshop*, pp. 43-52. IEEE Computer Society, Washington, DC, USA.

Karasavvas, K., Antonioletti, M., Atkinson, M., Hong, N. C., Sugden, T., Hume, A., Jackson, M., Krause, A. and Palansuriya, C. (2005). Introduction to OGSA-DAI services, pp.1-12. In *Lecture notes in Computer Science, volume 3458*, Springer-Verlag, Germany.



- Krauter, K., Buyya, R. and Maheswaran, M. (2002). A taxonomy and survey of grid resource management systems for distributed computing. *Software: Practice and Experience*, 32(2):135-164.
- Kutten, S., Peleg, D. and Vishkin, U. (2001). Deterministic resource discovery in distributed networks. In *Proceedings of the thirteenth annual ACM symposium on Parallel algorithms and architectures*, pp. 77-83. ACM Press, New York, NY, USA.
- Law, C. and Siu, K. (2000). An  $O(\log n)$  randomized resource discovery algorithm. Available online <http://www.autoidlabs.org/uploads/media/MIT-AUTOID-TR-004.pdf>. Last accessed 23rd May 2007.
- Li, W., Xu, Z., Dong, F. and Zhang, J. (2002). Grid resource discovery based on a routing-transferring model. In *Proceedings of the 3rd international workshop on grid computing*, pp. 145-156. In Lecture notes in Computer Science, volume 2536, Springer-Verlag, Germany.
- Lican, H., Zhaohui, W. and Yunhe, P. (2003). Virtual and Dynamic Hierarchical Architecture for E-Science Grid. *International Journal of High Performance Computing Applications*, 17(3): 329-347.
- Ludwig, S. and Santen, P. V. (2002). A grid service discovery matchmaker based on ontology description. In *Proceedings of the international EuroWeb Conference, Oxford, UK*.
- Maheswaran, M. and Krauter, K. (2000). A Parameter-Based Approach to Resource Discovery in Grid Computing Systems. In *Proceedings of the first international workshop on grid computing*, pp. 363-385. In Lecture notes in Computer Science, volume 1971, Springer-Verlag, Germany.
- MammoGrid (2007). European federated mammogram database implemented on a grid structure. Website <http://mammogrid.vitamib.com>. Last accessed 5th April, 2007.

- Mastroianni, C., Talia, D. and Trunfio, P. (2003). Managing heterogeneous resources in data mining applications on grids using XML-based metadata. In *Proceedings of the International Parallel and Distributed Processing Symposium*, pp. 22-26. IEEE Computer Society, Washington, DC, USA.
- Mastroianni, C., Talia, D. and Trunfio, P. (2004). Metadata for managing grid resources in data mining applications. *Journal of Grid Computing*, 2(1): 85-102.
- McGuinness, D. L., Fikes, R., Rice, J. and Wilder, S. (2000). An environment for merging and testing large ontologies. In Cohn, A. G., Giunchiglia, F. and Selman, B. (editors), *Proceedings of the Seventh International Conference on Principles of Knowledge Representation and Reasoning (KR2000)*. Morgan Kaufmann Publishers, San Francisco, CA.
- Medical Dictionary for Regulatory Activities (2007). Website <http://www.meddrmsso.com/MSSOWeb/index.htm>. Last accessed 23rd May 2007.
- Mena, E., Illarramendi, A., Kashyap, V. and Sheth, A. (2000). OBSERVER: An approach for query processing in global information systems based on interoperation across pre-existing ontologies. *Distributed and Parallel Databases - An International Journal*, 8(2): 223-271.
- Microsoft. (2000). *UDDI Technical White Paper*. Available online [http://www.uddi.org/pubs/Iru\\_UDDI\\_Technical\\_White\\_Paper.pdf](http://www.uddi.org/pubs/Iru_UDDI_Technical_White_Paper.pdf). Last accessed 23rd May 2007.
- Mitra, P., Wiederhold, G. and Kersten, M. (2000). A graph-oriented model for articulation of ontology interdependencies. In *Proceedings Conference on Extending Database Technology 2000 (EDBT'2000)*, Konstanz, Germany.
- myGrid. (2007). Website <http://www.mygrid.org.uk>. Last accessed 23rd March, 2007.

- Naseer, A. and Stergioulas, L. K. (2006a). Discovering HealthGrid Services. In *Proceedings of the IEEE International Conference on Services Computing (SCC'06)*, pp. 301-306.
- Naseer, A. and Stergioulas, L. K. (2006b). Integrating grid and web services: A critical assessment of methods and implications to resource discovery. In *Proceedings of the 2nd Workshop on Innovations in Web Infrastructure*. Available online <http://www.wmin.ac.uk/~courtes/iwi2006/naseer.pdf>. Last accessed 27th July, 2007.
- Naseer, A. and Stergioulas, L. K. (2006c). Resource discovery in grids and other distributed environments: States of the art. *Multiagent and Grid Systems - An International Journal*, 2(2): 163-182.
- Naseer, A. and Stergioulas, L. K. (2007). Combining web services and grid services: practical approaches and implications to resource discovery. In *Securing web services: practical usage of standards and specifications*, eds. Periorellis, P., ISBN 1599046393 (Idea Group Inc, USA), Chapter 12.
- National Digital Mammography Archive. (2007). Website <http://www.hoise.com/primeur/05/articles/monthly/AE-PR-02-05-1.html>. Last accessed 3rd April, 2007.
- Noy, N. F. (2004). Semantic integration: a survey of ontology-based approaches. *SIGMOD Rec.* 33(4): 65-70.
- Noy, N.F. and Musen, M.A. (2000). PROMPT: Algorithm and tool for automated ontology merging and alignment. In *17th National Conference on Artificial Intelligence (AAAI-2000)*, Austin, TX, USA.
- Nozaki, K., Akiyama, T., Shimojo, S., Maeda, S. and Tamagawa, H. (2005). Integration of computational fluid dynamics and computational aero acoustics on grid for dental applications. In *Proceedings of the 18th IEEE Symposium on Computer-Based Medical Systems*, pp. 517-522. IEEE Computer Society, Washington, DC, USA.

- Piggott, D., Teljeur, C. and Kelly, A. (2004). Exploring the potential for using the grid to support health impact assessment modelling. *Parallel Computing*, 30(9-10): 1073-1091.
- Power, D., Politou, E., Slaymaker, M., Harris, S. and Simpson, A. (2004). A relational approach to the capture of DICOM files for Grid-enabled medical imaging databases. In *Proceedings of the 2004 ACM symposium on applied computing*, pp.272-279. ACM Press, New York, NY, USA.
- Power, D. J., Politou, E. A., Slaymaker, M. A. and Simpson, A. C. (2006). Securing web services for deployment in health grids. *Future Generation Computer Systems*, 22(5): 547-570.
- Power, D., Slaymaker, M., Politou, E. and Simpson, A. (2005). Protecting sensitive patient data via query modification. In *Proceedings of the 2005 ACM symposium on applied computing*, pp.224-230. ACM Press, New York, NY, USA.
- Ripeanu, M. (2001). Peer-to-peer architecture case study: Gnutella network. In *Proceedings of the first international conference on Peer-to-Peer Computing*, pp. 99-100. IEEE Computer Society, Washington, DC, USA.
- Rowstron, A. I. T. and Druschel, P. (2001). Pastry: scalable, decentralized object location, and routing for large-scale peer-to-peer systems. In *Proceedings of the IFIP/ACM International Conference on Distributed Systems Platforms*, pp. 329. In Lecture notes in Computer Science, volume 2218, Springer-Verlag, Germany.
- Ruotsalainen, P. (2004). A cross-platform model for secure electronic health record communication. *International Journal of Medical Informatics*, 73(3): 291-295.
- Saroiu, S., Gummadi, P. K. and Gribble, S. D. (2002). A measurement study of peer-to-peer file sharing systems. In *Proceedings of the Multimedia Computing and Networking*. Available online [www.cs.toronto.edu/~stefan/publications/mmcn/2002/mmcn.html](http://www.cs.toronto.edu/~stefan/publications/mmcn/2002/mmcn.html). Last accessed 23rd May 2007.

- Scheres, S. H. W., Merino, A. J., Sorzano, C. O. S. and Carazo, J. M. (2005). Grid computing in 3D-EM image processing using Xmipp. *In Proceedings of the 18th symposium on computer-based medical system*, pp. 561-563. IEEE Computer Society, Washington, DC, USA.
- Semantic grid project. (2007). Website <http://www.semanticgrid.org/>. Last accessed 25th May 2007.
- Shiloach, Y. and Vishkin, U. (1982). An  $O(\log n)$  Parallel Connectivity Algorithm. *Journal of Algorithms*, 3(1): 57-67.
- Silva, J. S. and Ball, M. J. (2002). Prognosis for year 2013. *International Journal of Medical Informatics*, 66(1-3): 45-49.
- Simpson, A., Power, D., Slaymaker, M. and Politou, E. (2005). GIMI: generic infrastructure for medical informatics. *In Proceedings of the 18th symposium on computer-based medical system*, pp. 564-566. IEEE Computer Society, Washington, DC, USA.
- Slaughter, L. A., Soergel, D. and Rindflesch, T. C. (2006). Semantic representation of consumer questions and physician answers. *International Journal of Medical Informatics*, 75(7): 513-529.
- Sloot, P. M. A., Tirado-Ramos, A., Altintas, I., Bubak, M. and Boucher, C. A. (2006). From molecule to man: decision support in individualized e-health. *IEEE Computer*, 39(11): 40-46.
- Sorzano, C. O. S., Marabini, R., Velázquez-Muriel, J., Bilbao-Castro, J. R., Scheres, S. H. W., Carazo, J. M. and Pascual-Montano, A. (2004). XMIPP: A new generation of an open-source image processing package for electron microscopy. *Journal of Structural Biology*, 148(2): 194-204.
- Sotomayor, B. (2007). The Globus Toolkit 4 programmer's tutorial. Available online [http://gdp.globus.org/gt4-tutorial/singlehtml/progtutorial\\_0.2.1.html](http://gdp.globus.org/gt4-tutorial/singlehtml/progtutorial_0.2.1.html). Last accessed 23rd May 2007.

- Stevens, R., McEntire, R., Goble, C., Greenwood, M., Zhao, J., Wipat, A. and Li, P. (2004). myGrid and the drug discovery process. *Drug Discovery Today: BIOSILICO*, 2(4): 140-148.
- Stewart, C. A. (2004). Introduction to the edition. *Communications of the ACM*, 47(11): 30-33.
- Stoica, I., Morris, R., Liben-Nowell, D., Karger, D. R., Kaashoek, M. F., Dabek, F. and Balakrishnan, H. (2003). Chord: a scalable peer-to-peer lookup protocol for Internet applications. *IEEE/ACM Transactions on Networking*, 11(1): 17-32.
- Stumme, G. and Mädche, A. (2001). FCA-Merge: Bottom-up merging of ontologies. In *Proceedings of the 7th International Conference on Artificial Intelligence (IJCAI '01)*, pp. 225-230, Seattle, WA.
- Sycara, K., Klusch, M., Widoff, S. and Lu, J. (1999). Dynamic service matchmaking among agents in open information environments. *SIGMOD Record (ACM Special Interests Group on Management of Data)*, 28(1): 47-53.
- Talia, D. (2002). The Open Grid Services Architecture: where the grid meets the Web. *Internet Computing*, 6(6): 67-71.
- Tan, H., Chen, X. and Gu, J. (2007). Semanteme-based processing mechanism under data grid environment. *Journal of Huazhong University of Science and Technology (Natural Science Edition)*, 35(4): 22-25.
- Tangmunarunkit, H., Decker, S. and Kesselman, C. (2003). Ontology-Based Resource Matching in the Grid - The Grid Meets the Semantic Web, pp. 706-721. In *Lecture notes in Computer Science, volume 2870*, Springer-Verlag, Germany.
- TeraGrid Project . (2007). Website <http://www.teragrid.org>. Last accessed 25th May 2007.
- The University of Edinburgh. (2006). OGSA-DAI WSRF 2.2 user guide. Available online <http://www.ogsadai.org.uk/documentation/ogsadai-wsrf-2.2/doc/>. Last accessed 9th May, 2007.

- Thompson, D. (2000). Understanding LDAP, White Paper. *Microsoft Corporation*.
- Tirado-Ramos, A., Groen, D. and Sloot, P. (2005). On-line application performance monitoring of blood flow simulation in computational grid architectures. In *Proceedings of the 18th symposium on computer-based medical system*, pp. 511-516. IEEE Computer Society, Washington, DC, USA.
- Tohsato, Y., Kosaka, T., Date, S., Shimojo, S. and Matsuda, H. (2005). Heterogeneous database federation using grid technology for drug discovery process, pp. 43-52. In *Lecture notes in Computer Science, volume 3370*, Springer-Verlag, Germany.
- Tripp, S. and Bichelmeyer, B. (1990). Rapid prototyping: An alternative instructional design strategy. *Educational Technology Research and Development*, 38(1): 31-44.
- Tuecke, S., Czajkowski, K., Foster, I., Frey, J., Graham, S., Kesselman, C., Maguire, T., Sandholm, T., Snelling, D. and Vanderbilt, P. (2003). Open Grid Services Infrastructure (OGSI), Global Grid Forum (GGF). Available online <http://www.globus.org/alliance/publications/papers.php#GSSpec>. Last accessed 27th July, 2007.
- Twardoch, A. (2003). Web Services Technologies: XML, SOAP and UDDI. Available online <http://www.twardoch.com/webservices/twwebsrv.pdf>. Last accessed 27th July, 2007.
- Vadhiyar, S. S. and Dongarra, J. J. (2005). Self adaptivity in grid computing. *Concurrency and Computation: Practice and Experience*, 17(2-4): 235-257.
- Verschelde, J-L., Dos Santos, M. C., Deray, T., Smith, B. and Ceusters, W. (2004). Ontology-Assisted Database Integration to Support Natural Language Processing and Biomedical Data-mining. *Journal of Integrative Bioinformatics*, 1(1): 1-10. Available online [http://journal.imbio.de/index.php?paper\\_id=1](http://journal.imbio.de/index.php?paper_id=1). Last accessed 27th July, 2007.

- Web Services Distributed Management. (2006). *Defining a Web services architecture to manage distributed resources*. Available online [http://www.oasis-open.org/committees/tc\\_home.php?wg\\_abbrev=wsdm](http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsdm). Last accessed 25th May, 2007.
- Weiss, M., Esfandiari, B. and Luo, Y. (2007). Towards a classification of web service feature interactions. *Computer Networks*, 51(2): 359-381.
- Whiddett, R., Hunter, I., Engelbrecht, J. and Handy, J. (2006). Patients' attitudes towards sharing their health information, *International Journal of Medical Informatics*, 75(7): 530-541.
- Wu, B., Dovey, M., Ng, M. H., Tai, K., Murdock, S., Jeffreys, P., Cox, S., Essex, J. and Sansom, M. S. P. (2004). A web / grid portal implementation of BioSimGrid: A biomolecular simulation database. *In Proceedings of the Information Technology: Coding and Computing*, pp. 50-54.
- Xing, L., Wang, J. and Zhou, B. (2006). Integration and access of data resources in simulation grid system. *Journal of System Simulation*, 18(2): 369-372.
- Yang-Suk, K., Casanova, H. and Chien, A. A. (2004). Realistic modeling and synthesis of resources for computational grids. *In Proceedings of the 2004 supercomputing*, pp. 54-63. IEEE Computer Society, Washington, DC, USA.
- Zamboulis, L., Fan, H., Belhajjame, K., Siepen, J., Jones, A., Martin, N., Poulouvassilis, A., Hubbard, S., Embury, S. M. and Paton, N. W. (2006). Data access and integration in the ISPIDER proteomics Grid. Available online <http://www.cs.man.ac.uk/~norm/papers/dils06.pdf>. Last accessed 25th May, 2007.
- Zang, T., Jie, W., Hung, T., Lei, Z., Turner, S. J. and Cai, W. (2004). The design and implementation of an OGSA-based grid information service. *In Proceedings of the IEEE web services*, pp. 566-573. IEEE Computer Society, Washington, DC, USA.



- Zhao, B.Y., Kubiawicz, J.D. and Joseph, A.D. (2001). Tapestry: An Infrastructure for Fault-tolerant. *Technical report: CSD-01-1141*. UC Berkeley.
- Zhou, J. and Wang, M. (2006). *Semantic integration of enterprise information: Challenges and basic principles*. In *Proceedings of the ASWC 2006*, pp. 219-233.
- Zhu, C., Liu, Z., Zhang, W., Xiao, W., Xu, Z. and Yang, D. (2004). Decentralized grid resource discovery based on resource information community. *Journal of Grid Computing*, 2(3): 261-277.
- Zhuchkov, A., Tverdokhlebov, N. and Kravchenko, A. (2006). Advancing of Russian ChemBioGrid by bringing data management tools into collaborative environment. *Studies in health technology and informatics*, 120: 179-186.
- Zhuge, H. (2004a). Semantics, Resource and Grid (editorial to the special issue on semantic grid and knowledge grid: the next-generation web). *Future Generation Computer Systems*, 20(1): 1-5.
- Zhuge, H. (2004b). Resource space model, its design method and applications. *Journal of Systems and Software*, 72(1): 71-81.

## Appendix-A

### Downloads

In order to set up the deployment environment, there was a need to make necessary installations of various Grid technologies such as GT4, and OGSA-DAI and construct heterogeneous data sources (experimental databases). Hence the core Grid services from the GT4 toolkit and OGSA-DAI (OGSA-DAI WSRF 2.2) were installed. Both of these Grid technologies are open-source and are available for download from their respective websites:

GT4: <http://www.globus.org/toolkit/downloads/4.0.4/> (Figure 50)

OGSA-DAI: <http://www.ogsadai.org.uk/downloads/> (Figure 51)

For developing the prototype, two LINUX machines (with Fedora Core 6.0 installed) were used.

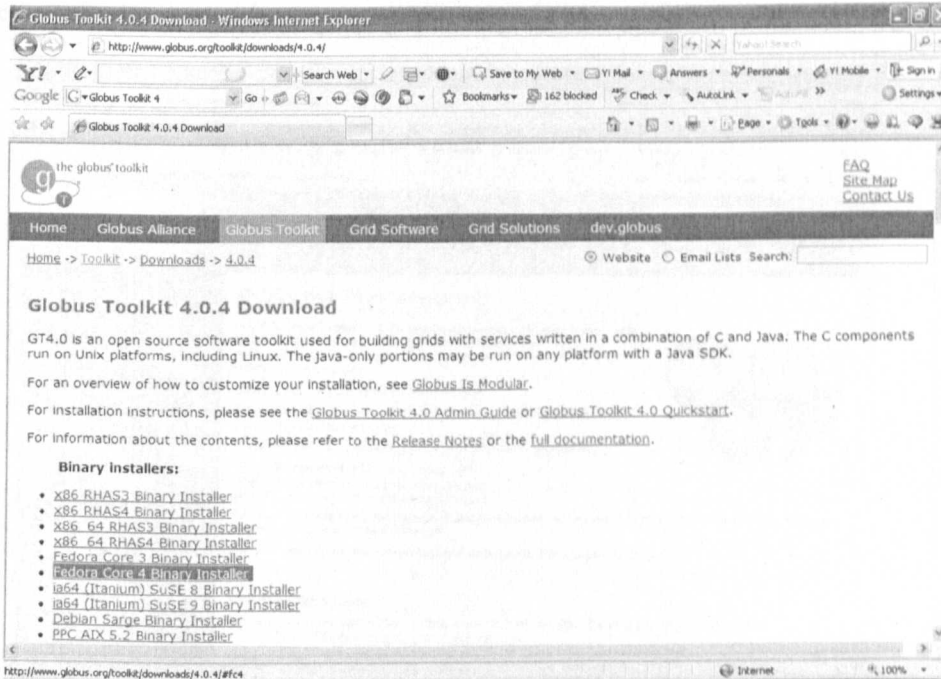


Figure 50: Globus Webpage from where GT4.0.4 was downloaded

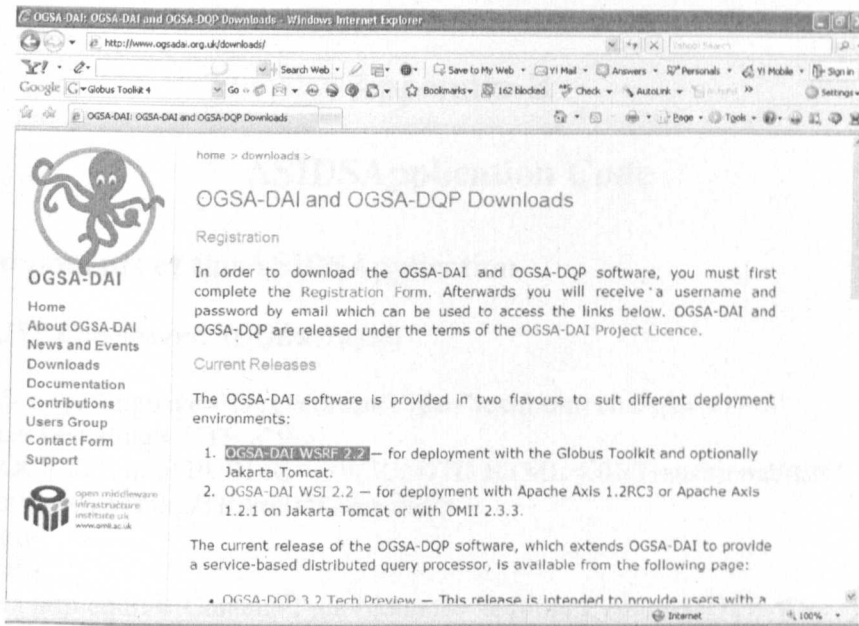


Figure 51: OGSA-DAI Webpage from where OGSA-DAI WSRF 2.2 was downloaded

MySQL GUI Tools were used for creating and managing the data sources. These are open-source and can be downloaded from the website (Figure 52). A set of MySQL GUI Tools such as MySQL Administrator and MySQL Query Browser are very useful while creating and populating the data sources.

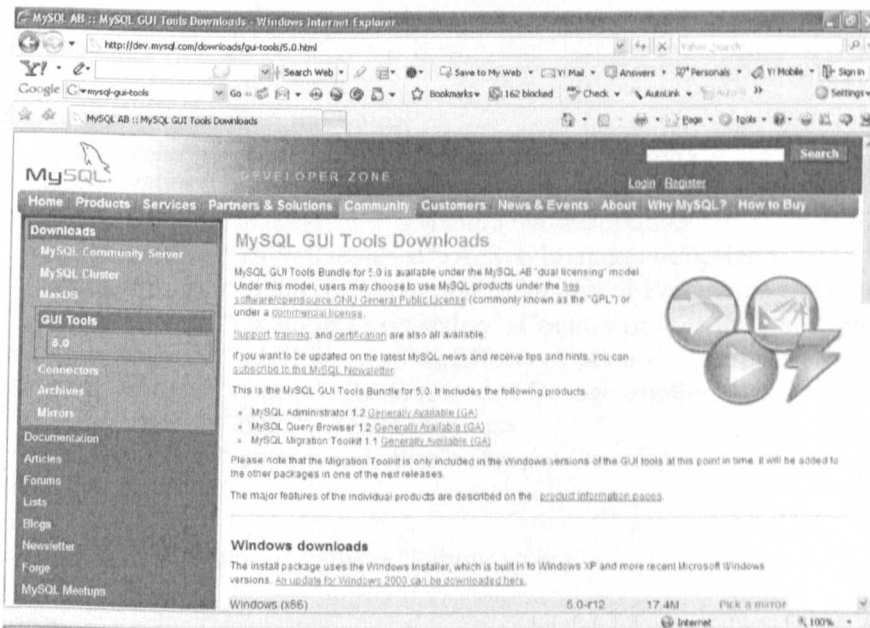


Figure 52: MySQL Webpage from where MySQL GUI Tools were downloaded

## Appendix-B

### ASIDSApplication Code

#### Components of the ASIDSApplication

##### a) JSP Component: (QQDuery.jsp)

```

<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Servlet Running - Data Discovery Resultset</title>
</head>
<body>

<br><br><br>

<form action = "ResultServlet" method = "POST">
    Please entre your keyword: <input type = "text" name = "userkeyword" size
= "25">
    and select your options from the drop down menu
<br><br><br>

<select name = "menulist" multiple>
    <option value = "medicineName">Drug Name</option>
    <option value = "batchNumber">Drug Batch Number</option>
    <option value = "manufacurer">Manufacturer</option>
    <option value = "activeSubstance">Active Ingredient</option>
    <option value = "countryofProduction">Country of Production</option>
    <option value = "countryofPrescription">Country of Distribution</option>
    <option value = "sideEffects">Side Effects</option>
    <option value = "dosage">Recommended Dosage</option>
    <option value = "aDRs">ADRs</option>
    <option value = "recommendedFor">Disease</option>
</select>

    <input type = "submit" value = "Submit Query">

</form>
</body>
</html>

```

**b) Servlet Component:      DataDiscoveryClient.java**

```

package brunel.ogsadai.test;

import java.io.IOException;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.io.*;
import java.util.Map;

/**
 * Servlet implementation class for Servlet: DataDiscoveryClient
 */
public class DataDiscoveryClient extends javax.servlet.http.HttpServlet
implements javax.servlet.Servlet {
    /* (non-Java-doc)
     * @see javax.servlet.http.HttpServlet#HttpServlet()
     */
    public DataDiscoveryClient() {
        super();
    }

    /* (non-Java-doc)
     * @see javax.servlet.http.HttpServlet#doGet(HttpServletRequest request,
    HttpServletResponse response)
     */
    protected void doGet(HttpServletRequest request, HttpServletResponse
    response) throws ServletException, IOException {
        // TODO Auto-generated method stub
    }

    /* (non-Java-doc)
     * @see javax.servlet.http.HttpServlet#doPost(HttpServletRequest request,
    HttpServletResponse response)
     */
    protected void doPost(HttpServletRequest request, HttpServletResponse
    response) throws ServletException, IOException {
        // TODO Auto-generated method stub

        response.setContentType("text/html;charset=UTF-8");
        PrintWriter out = response.getWriter();

        out.println("<br><h1> Query Results: </h1><br>");
        out.println("<b> Your query generated the results as
below:</b><br>");

```

```

        String[] serviceHandle =
        {"http://localhost:8080/wsrf/services/ogsadai/DataDiscoveryDisc",
        "http://134.83.86.129:8080/wsrf/services/ogsadai/DataDiscovery1"};
        String[] dsrID = {"DSR_DS1", "DISC_DSR_DS1"};

//        String handle =
        "http://localhost:8080/wsrf/services/ogsadai/DataDiscoveryDisc";
//        String id = "DISC_DSR_DS1";
//        String handlelaptop =
        "http://134.83.86.129:8080/wsrf/services/ogsadai/DataDiscovery1";
//        String idlaptop = "DSR_DS1";

        String userKeyword =
        request.getParameter("userkeyword").toString();
        String menuItem = request.getParameter("menulist").toString();
        String userQuery = "initQueryValue";
        String queryResult = "initResultValue";

        if( (menuItem == null) || (userKeyword == null) ){
            userKeyword = "Aspiritab";
            menuItem = "medicineName";
        }

        // Send multiple handles

        // Going to make a new object
        for(int k = 0; k < serviceHandle.length; k++){
            Mapping metaObj = new
            Mapping(serviceHandle[k],dsrID[k]);

            //Getting DSResource IDs
            String[] DSR_IDs = metaObj.getRIdsNames();
            out.println(Mapping.print_RID_Array(DSR_IDs));

            //Constructing Query
            for (int i = 0; i < DSR_IDs.length; i++){
                Map colMap =
                metaObj.getMapOfColumns(DSR_IDs[i]);
                Map tableMap =
                metaObj.getMapOfTables(DSR_IDs[i]);

                //Semantic Ontology Mapping for Key-Value Pairs
                String menuItem = "medicineName";
                String valMatch =
                Mapping.matchColMapValue(colMap, menuItem);
                out.println("<br> Column Name: " + valMatch);
            }
        }

```

```

        //System.out.println(tableNameValueMatch);

        for (int j = 0; j < tableMap.size(); j++){
            String tableNameValueMatch =
Mapping.matchTabMapValue(tableMap);
            out.println("<br> Table Name: " +
tableNameValueMatch);

            userQuery = "select * from " +
tableNameValueMatch + " where " + valMatch + " like " + userKeyword + "%";
            queryResult =
metaObj.performQuery(DSR_IDs[i], userQuery);
            out.println(queryResult);
        }
    }
    out.close();
}
}

```

**c) Semantic Mapping Component: Mapping.java**

```
package brunel.ogsadai.test;
```

```
//Data Discovery Class -- 02/04/2007
```

```
 //(c) School of Information Systems, Computing & Mathematics, 2007.
```

```
 //(c) Brunel University, 2007.
```

```
//import java.util.ArrayList;
```

```
import java.util.HashMap;
```

```
import java.util.Iterator;
```

```
import java.util.List;
```

```
import java.util.Map;
```

```
import javax.xml.namespace.QName;
```

```
import org.w3c.dom.Element;
```

```
import uk.org.ogsadai.client.toolkit.GenericServiceFetcher;
```

```
import uk.org.ogsadai.client.toolkit.ResourceID;
```

```
import uk.org.ogsadai.client.toolkit.activity.ActivityRequest;
```

```
import uk.org.ogsadai.client.toolkit.activity.delivery.DeliverFromURL;
```

```
import uk.org.ogsadai.client.toolkit.activity.sql.SQLQuery;
```

```
import uk.org.ogsadai.client.toolkit.activity.sql.WebRowSet;
```

```
import uk.org.ogsadai.client.toolkit.activity.transform.XSLTransform;
```

```
import uk.org.ogsadai.client.toolkit.properties.Property;
```

```
import uk.org.ogsadai.client.toolkit.service.DataService;
```

```
import uk.org.ogsadai.common.xml.XMLUtilities;
```

```

/**
 * A Class for Mapping the Column and Tables from the DS-Resource
 * Configuration file for Metadata
 *
 * @author Aisha Naseer.
 */

public class Mapping {
    DataService service = null;
    Map id_service = new HashMap();

    String handle = null;
    String id = "DISC_DSR_DS3";

    static String COLUMN_MAPPING_PROPERTY =
"{http://ogsadai.org.uk/namespaces/2005/10/config}columnMapping";
    static String TABLE_MAPPING_PROPERTY =
"{http://ogsadai.org.uk/namespaces/2005/10/config}tableMapping";

    /**
     * @param args
     */

    public Mapping (String handle, String firstRID){
        System.out.println("Constructor going in try");
        try{
            this.handle = handle;
            DataService tempDS = createService(firstRID);
            this.service = tempDS;
            String [] DSR_IDs = getRIdsNames();

            for(int i = 0; i < DSR_IDs.length; i++){
                tempDS = createService(DSR_IDs[i]);
                id_service.put(DSR_IDs[i], tempDS);
            }
        } catch (Exception e) {
            // TODO: handle exception
            System.out.println("Cannot instantiate Constructor");
        }
    }

    public DataService getDataService (String DSR_IDs){
        DataService tempDS = (DataService)id_service.get(DSR_IDs);
        return tempDS;
    }
}

```



```

public DataService createService(String RID){
    DataService tempDS = null;

    try{
        tempDS =
GenericServiceFetcher.getInstance().getDataService(this.handle, RID);
    } catch (Exception e){
        e.printStackTrace();
    }
    return tempDS;
}

public String[] getRidsNames(){
    ResourceID[] resourceIDs = null;
    String[] DSR_IDs = null;

    try{
        resourceIDs = getResources();
        DSR_IDs = new String[resourceIDs.length];

        for (int i = 0; i < resourceIDs.length; i++) {
            DSR_IDs[i] = (resourceIDs[i].getName());
        }
    } catch (Exception e){
        e.printStackTrace();
    }
    return DSR_IDs;
}

public ResourceID[] getResources(){
    ResourceID[] resourceIDs = null;

    try{
        resourceIDs = service.getResourceIDs();
    } catch (Exception e){
        e.printStackTrace();
    }
    return resourceIDs;
}

public Map getMapOfColumns(String DSR_IDs){
    Property DSR_Property = null;
    try{
        DataService tempDS = getDataService(DSR_IDs);
        DSR_Property =
tempDS.getProperty(QName.valueOf(COLUMN_MAPPING_PROPERTY));

```

```

        if (DSR_Property == null){
            System.out.println("No
COLUMN_MAPPING_PROPERTY property in DSR_ID: " + DSR_IDs);
            return null;
        }
    } catch (Exception e){
        //e.printStackTrace();
        System.out.println("No property in DSR_ID: " + id);
        return null;
    }

    Map colMap = new HashMap();
    Object propValue = DSR_Property.getValue();

    List list = XMLUtilities.getChildElements((Element)propValue);

    Iterator iter = list.iterator();
    while(iter.hasNext()){
        Element element = (Element)iter.next();

        String key = element.getLocalName();
        String value =
        (String)XMLUtilities.getTextContent(element);

        colMap.put(key, value); //fills in the Map with key, values
pairs
    }
    return colMap;
}

public static String getColMapKey(Map colMap){
    String colMapKey = "";
    if (colMap == null)
        return "No Map found";

    Iterator iter = colMap.entrySet().iterator();

    while(iter.hasNext()){
        Map.Entry colMapEntry = (Map.Entry)iter.next();
        String ontologyKey = (String)colMapEntry.getKey();
        colMapKey += ontologyKey;
    }
    return colMapKey;
}

public static String getColMapValue(Map colMap){
    String colMapValue = "";

```

```

        if (colMap == null)
            return "No Map found";

        Iterator iter = colMap.entrySet().iterator();

        while(iter.hasNext()){
            Map.Entry colMapEntry = (Map.Entry)iter.next();
            String colValue = (String)colMapEntry.getValue();
            colMapValue += colValue;
        }
        return colMapValue;
    }

    public static String matchColMapValue(Map colMap, String menuItem){
        String valueMatch = "initVal";
        if (colMap == null)
            return "No Map found";

        Iterator iter = colMap.entrySet().iterator();

        while(iter.hasNext()){
            Map.Entry colMapEntry = (Map.Entry)iter.next();

            String ontologyKey = (String)colMapEntry.getKey();
            valueMatch = colMap.get(ontologyKey).toString();
            String menuItem = "manufacurer";
            if(ontologyKey.equals(menuItem)){
                System.out.println("Key: " + ontologyKey);
                //System.out.println("Match_Val: "+ valueMatch);
                return valueMatch;
            }
            else
                valueMatch = null;
        }
        return null;
    }

    public Map getMapOfTables(String DSR_IDs){
        Property DSR_Property = null;
        try{
            DataService tempDS = getDataService(DSR_IDs);
            DSR_Property =
tempDS.getProperty(QName.valueOf(TABLE_MAPPING_PROPERTY));
            if (DSR_Property == null){
                System.out.println("No
TABLE_MAPPING_PROPERTY property in DSR_ID: " + DSR_IDs);
                return null;
            }
        }
    }

```

```

    }
    } catch (Exception e){
        //e.printStackTrace();
        System.out.println("No property in DSR_ID: " + id);
        return null;
    }

    Map colMap = new HashMap();
    Object propValue = DSR_Property.getValue();

    List list = XMLUtilities.getChildElements((Element)propValue);

    Iterator iter = list.iterator();
    while(iter.hasNext()){
        Element element = (Element)iter.next();

        String key = element.getLocalName();
        String value =
        (String)XMLUtilities.getTextContent(element);

        colMap.put(key, value); //fills in the Map with key, values
pairs
    }
    return colMap;
}

public static String matchTabMapValue(Map tableMap){
    String tableNameValueMatch = "initVal";
    if (tableMap == null)
        return "No Map found";

    Iterator iter = tableMap.entrySet().iterator();

    while(iter.hasNext()){
        Map.Entry colMapEntry = (Map.Entry)iter.next();

        String ontologyKey = (String)colMapEntry.getKey();
        tableNameValueMatch =
tableMap.get(ontologyKey).toString();
        return tableNameValueMatch;
    }
    return null;
}

public String performQuery(String id, String sqlQuery) {
    String result = "init preform";

```

```

    try{
        DataService service = getDataService(id);

        String url = "http://localhost:8077/t.xml";

        DeliverFromURL deliver = new DeliverFromURL(url);
        SQLQuery query = new SQLQuery(sqlQuery);
        WebRowSet rowset = new
WebRowSet(query.getOutputStream());

        // Construct the transformation activity
        XSLTransform transform = new XSLTransform();
        transform.setXMLInput(rowset.getOutputStream());
        transform.setXSLTInput(deliver.getOutputStream());

        // Construct the request
        ActivityRequest request = new ActivityRequest();
        request.add(deliver);
        request.add(query);
        request.add(rowset);
        request.add(transform);

        // Performing request
        service.perform(request);
        result = transform.getOutputStream().getData();
    } catch (Exception e) {
        // TODO: handle exception
        e.printStackTrace();
    }
    return result;
}

public static String print_RID_Array(String[] DSR_IDs){
    String DSRs = "Data Service Resources: ";
    for (int i = 0; i < DSR_IDs.length; i++){
        DSRs += DSR_IDs[i] + ",\t";
    }
    return DSRs;
}

public static void main(String[] args) {
    // TODO Auto-generated method stub
}
}

```